CLASSIFICAÇÃO DE NAVIOS BASEADA EM CURVAS PRINCIPAIS

Helena Lopez Fernandez

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Aprovada por:

Prof. José Manoel de Seixas, D.Sc.

Prof. Luiz Pereira Calôba, Dr. Ing.

Prof. Eduardo Antônio Barros da Silva, Ph.D.

Prof. Marcelo Portes de Albuquerque, D.Sc.

Dr. Sergio Rodrigues Neves, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

MARÇO DE 2005

FERNANDEZ, HELENA LOPEZ

Classificação de navios baseada em curvas principais [Rio de Janeiro] 2005

XVI, 171 pp 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia Elétrica, 2005)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1.Classificação de navios 2.Curvas principais 3.Processamento de sinais de sonar passivo 4.Processamento de imagens infravermelhas 5.Processamento estatístico de sinais

I.COPPE/UFRJ II.Título (série)

- A Deus, pela saúde e disposição que me permitiram a realização deste trabalho.
- Ao meu marido muito querido, Carlos, pelo seu amor, auxílio, paciência e compreensão, sem os quais nem uma pequena parte deste trabalho teria sido realizada.
- Aos meus queridos filhos, Mariana e Carlos Eduardo, pelo incentivo e compreensão, fundamentais durante todo o período de curso.
- Aos meu orientador, José Manoel de Seixas, por toda ajuda e amizade, indispensáveis na elaboração deste trabalho.
- Aos meus queridos pais, Manuel e Maria del Carmen, minha avó, Gracia, e meus irmãos, Eduardo, Isabel e Juan Carlos, pelo seu carinho.
- Aos meus sogros e cunhados, pelo apoio incondicional.
- Aos estimados amigos Márcio Rodrigues Pereira da Silva e João Batista de Oliveira Souza Filho pela ajuda e incentivo durante a elaboração deste trabalho.
- Ao meu orientador técnico, o pesquisador Sérgio Rodrigues Neves e ao meu colega Jorge Amaral Alves, pelo incentivo e orientação.
- Por fim, aos professores e colegas do LPS, por todo o apoio recebido.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

CLASSIFICAÇÃO DE NAVIOS BASEADA EM CURVAS PRINCIPAIS

Helena Lopez Fernandez

Março/2005

Orientador: José Manoel de Seixas

Programa: Engenharia Elétrica

Este trabalho propõe a classificação de navios baseada em curvas principais, uma técnica de procesamento estatístico empregada na representação de dados em um espaço de dimensão elevada. Dados experimentais obtidos por um sistema de sonar passivo e imagens infravermelhas simuladas são empregadas na avaliação do sistema classificador proposto.

Sinais acústicos emitidos por navios pertencentes a oito classes, em diferentes condições operativas, são captados pelo sistema de sonar passivo. Esses sinais, convenientemente pré-processados sob a forma de espectrogramas, foram empregados na extração de curvas principais, as quais formam a base do classificador proposto. Considerando como critério de decisão a menor distância euclidiana do espectrograma recebido à curva extraída para cada classe, é obtida uma eficiência média de classificação igual a 96,8%.

O segundo sistema identifica navios a partir de imagens infravermelhas segmentadas. O conjunto de dados empregado é composto por 40.600 vistas de navios, de cinco classes diferentes, geradas a partir de modelos tridimensionais. A informação discriminante foi obtida a partir do mapeamento dos contornos externos das vistas, tendo sido obtida uma eficiência média de classificação igual a 97,3%, valor este significativamente melhor que o obtido empregando-se um classificador neural baseado em momentos invariantes.

A utilização da técnica de curvas principais na solução dos problemas de classificação abordados resultou em classificadores eficientes, escaláveis, facilmente configuráveis e de baixo custo computacional na fase de operação, o que torna atraente a implementação dos sistemas propostos.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

SHIP CLASSIFICATION BASED ON PRINCIPAL CURVES

Helena Lopez Fernandez

March/2005

Advisor: José Manoel de Seixas

Department: Electrical Engineering

This work proposes ship classification based on principal curves, a statistical processing technique used to represent a high-dimensional data space. Experimental data acquired by passive sonar and infrared images are used to verify the proposed classifier system's performance.

Acoustic signals radiated by ships belonging to eight classes, in different operative conditions, were detected by a passive sonar. These signals, conveniently pre-processed in the form of spectrograms, were employed in the extraction of principal curves, which form the kernel of the proposed classifier. Considering as the decision criterion the smallest Euclidean distance from curves assigned to classes for an incoming spectrogram, a mean classification efficiency of 96,8% was attained.

The second system identifies ships from segmented infrared images. The data set used is composed of 40,600 ship views, generated from multifaceted patches models. The discriminant information was obtained mapping each view's external contours. A mean classification efficiency equal to 97,3% was achieved, outperforming a neural network classifier based on moment invariants.

The principal curves tecnique made it possible to obtain efficient, scalable and easily configurable classifiers, which present, also, a low computational cost during the operational phase. Those features make the systems implementation attractive.

Sumário

1 Introdução **1**

1.1 Motivação 2

1.2 O que foi feito 4

1.3 Organização do trabalho 5

2 Curvas Principais **7**

2.1 Curvas principais auto-consistentes 8

2.1.1 Definição de HS 9

2.1.2 Algoritmo proposto por HS 15

2.1.3 Problemas potenciais do modelo de HS 18

2.1.4 Correção da tendência de estimação 22

2.2 Definições alternativas 24

2.2.1 Modelos paramétricos 25

2.2.2 Curvas principais ajustadas 27

2.2.3 O modelo de Delicado 29

2.2.4 Algoritmo de k-segmentos não-suave 30

2.2.5 Análise comparativa dos métodos de extração de curvas principais 37

2.3 Aplicações 38

2.4 Sumário 40

3 Classificação de navios baseada em sinais de sonar passivo **42**

3.1 Introdução 43

3.2 O conjunto de dados 50

3.2.1 Análise estatística dos dados 55

3.2.2 Distâncias entre os centros das classes 58
3.2.3 Distâncias entre os espectros das classes 62
3.3 Implementação 63
3.3.1 O classificador baseado em curvas principais 63
3.3.2 O algoritmo de k-segmentos não-suave 64
3.3.3 Extração das curvas principais 66
3.3.4 Quantidade de segmentos e comprimentos das curvas 68
3.3.5 Distâncias entre espectros e curvas 71
3.3.6 Distâncias entre curvas 72
3.4 Resultados de classificação 76
3.5 Comparação do desempenho da classificação 80
3.6 Sumário 82

4 Classificação de navios baseada em imagens infravermelhas 83
4.1 Introdução 84
4.2 O conjunto de dados 88
4.3 Extração de características discriminantes 96
4.3.1 Mapeamento do tipo I 98
4.3.2 Mapeamento do tipo II 100
4.3.3 Mapeamento do tipo III 101
4.3.4 Mapeamento do tipo IV 102
4.3.5 Mapeamento do tipo V 102
4.4 Implementação 104
4.4.1 O classificador de vistas de navios baseado em curvas principais 105
4.5 Resultados 106
4.5.1 Classificador I 107
4.5.2 Classificador II 110
4.5.3 Classificador III 113
4.5.4 Classificador IV 114
4.5.5 Classificador V 118
4.5.6 Análise do desempenho do classificador para dados ruidosos . 121
4.5.7 Análise comparativa do desempenho dos classificadores e considerações práticas 122

4.6 Sumário 125

5 Conclusão **127**

5.1 Possíveis Trabalhos Futuros 129

Referências Bibliográficas **132**

A Códigos-fonte **141**

A.1 Classificador de navios baseado em sinais de sonar passivo 141

A.1.1 Rotina de classificação implementada em linguagem C 141

A.1.2 Rotina de classificação implementada em Matlab6 143

A.2 Classificador de navios baseado em imagens 145

A.2.1 Rotina de classificação implementada em linguagem C 145

A.2.2 Rotina de classificação implementada em Matlab6 147

A.2.3 Arquivos de modelos das classes de navios 148

A.2.4 Rotina de geração das vistas sintéticas 158

A.2.5 Rotina de mapeamento do tipo I 160

A.2.6 Rotina de mapeamento do tipo II 161

A.2.7 Rotina de mapeamento do tipo III 163

A.2.8 Rotina de mapeamento do tipo IV 165

A.2.9 Rotina de mapeamento do tipo V 168

Lista de Figuras

2.1 Algumas formas de representação de dados: (a) regressão linear; (b) primeira componente principal; (c) regressão não-linear; e (d) curva principal (extraída de [1]). 8

2.2 Modelo de uma curva principal. 10

2.3 Curva principal obtida para uma distribuição de dados. 11

2.4 Distância de um ponto a uma curva poligonal. 13

2.5 Curva principal obtida para um conjunto finito de dados. Cada ponto $\mathbf{f}(t_i)$ da curva é definido como a média ponderada de pontos próximos de $\mathbf{x}_i$ numa vizinhança cujo tamanho é definido por um coeficiente de suavidade. 18

2.6 Modelo para estudo da tendência devida ao modelo na curva principal definida por HS. 20

2.7 Modelo para estudo da tendência devida à estimação na curva principal definida por HS. 21

2.8 Curvas principais obtidas com o algoritmo de HS para eventos gerados a partir de uma distribuição uniforme de pontos distribuídos sobre uma circunferência com ruído aditivo Gaussiano. Cada curva foi obtida empregando-se um valor distinto de coeficiente de suavidade: 0,2 no caso da curva mais externa (tracejada); 0,3 a curva central (pontilhada) e 0,5 a interna (linha cheia). (extraída de [2]). . . 23

2.9 Curvas principais obtidas com o algoritmo de HS modificado por BR empregando os mesmos dados e valores do coeficiente de suavidade (0,2 curva tracejada, 0,3 curva pontilhada e 0,5 curva cheia) da Figura 2.8 (extraída de [2]). 24

2.10 Curva principal obtida para dados gerados a partir de uma distribuição arbitrária empregando o algoritmo PLA. 29
2.11 Seqüência de construção de uma curva principal, para um conjunto de pontos gerados a partir de uma distribuição arbitrária, empregando o método de k-segmentos não-suave. As linhas grossas representam os segmentos obtidos pelo algoritmo enquanto que as linhas finas representam junções que unem os segmentos. A primeira aproximação da curva é mostrada em (a) e trata-se de um segmento de reta cuja direção coincide com a da primeira componente principal de todo o conjunto de dados. A segunda aproximação, mostrada em (b) trata-se de uma curva poligonal formada por dois segmentos unidos por uma junção. O restante da seqüência de construção é mostrada nos gráficos seguintes, resultando numa curva poligonal composta por oito segmentos, conforme mostrado em (h). 32
2.12 Fluxograma simplificado do algoritmo de k-segmentos não-suave. . . . 33

3.1 Obtenção dos ruídos irradiados por um navio durante uma corrida em uma raia acústica. 51
3.2 Cadeia de pré-processamento para obtenção dos espectros. 51
3.3 Janela empregada no algoritmo TPSW. 53
3.4 Amplitude do sinal antes do pré-processamento pelo algoritmo de TPSW (acima) e após o emprego do algoritmo de TPSW (abaixo). . 54
3.5 Espectrograma típico obtido a partir dos sinais adquiridos experimentalmente. 54
3.6 Índices dos dez tons de freqüência de maior intensidade, na média, irradiados por navios das diversas classes. O eixo dos x serve como índice de ordem de intensidade dos tons, assim, para $x = 1$ verifica-se, para todas as classe, os valores dos índices dos tons mais intenso, correspondente aos valores da primeira linha da Tabela 3.3. Analogamente, para $x = 2$, verificam-se os valores contidos na segunda linha da tabela citada. 56

3.7 Histograma e ajuste da fdp para o tom de freqüência de índice 12 da classe A. O ajuste é validado pelo teste de aderência do chi-quadrado, uma vez que o valor do χ^2 obtido é igual a 62,8 (para 49 graus de liberdade), inferior ao valor limite do χ^2, o qual é igual a 66,3. 58
3.8 Histograma e ajuste da fdp para o tom de freqüência de índice 12 da classe E. O ajuste é rejeitado pelo teste de aderência do chi-quadrado, uma vez que o valor do χ^2 obtido é igual a 325,7 (para 159 graus de liberdade), superior ao valor limite do χ^2, o qual é igual a 189,4. . . . 59
3.9 Exemplos de espectrogramas extraídos a partir de corridas de navios pertencentes à classe E. 60
3.10 Médias das distâncias euclidianas entre centros. 61
3.11 Médias das distâncias entre espectros. 63
3.12 Classificação de um evento. 65
3.13 Valores da função de distância empírica em função do número de segmentos das curvas de cada classe. A classe é indicada pelas letras junto às curvas. 69
3.14 Médias das distâncias entre espectros e curvas (unidade arbitrária). . 72

4.1 Imagem infravermelha de um navio de superfície. 85
4.2 Foto de um porta-aviões de forma e dimensões compatíveis com o modelo da classe. 91
4.3 Vistas da classe A (porta-aviões) para os seguintes ângulos de azimute (az) e elevação (el): (a) az = -90^o, el = 0^o; (b) az = -60^o, el = 15^o; az = -30^o, el = 15^o e az = 0^o, el = 0^o. 91
4.4 Foto de um contratorpedeiro de forma e dimensões compatíveis com o modelo da classe. 92
4.5 Vistas da classe B (contratorpedeiro) para os seguintes ângulos de azimute (az) e elevação (el): (a) az = -90^o, el = 0^o; (b) az = -60^o, el = 15^o; az = -30^o, el = 15^o e az = 0^o, el = 0^o. 92
4.6 Foto de uma fragata de forma e dimensões compatíveis com o modelo da classe. 93

4.7 Vistas da classe C (fragata) para os seguintes ângulos de azimute (az) e elevação (el): (a) az = -90^o, el = 0^o; (b) az = -60^o, el = 15^o; az = -30^o, el = 15^o e az = 0^o, el = 0^o. 93
4.8 Foto de um navio-patrulha de forma e dimensões compatíveis com o modelo da classe. 94
4.9 Vistas da classe D (navio-patrulha) para os seguintes ângulos de azimute (az) e elevação (el): (a) az = -90^o, el = 0^o; (b) az = -60^o, el = 15^o; az = -30^o, el = 15^o e az = 0^o, el = 0^o. 94
4.10 Foto de um navio-tanque de forma e dimensões compatíveis com o modelo da classe. 95
4.11 Vistas da classe E (navio-tanque) para os seguintes ângulos de azimute (az) e elevação (el): (a) az = -90^o, el = 0^o; (b) az = -60^o, el = 15^o; az = -30^o, el = 15^o e az = 0^o, el = 0^o. 95
4.12 Quadro contendo uma vista arbitrária e pontos de referência. 99
4.13 Mapeamento do tipo I: (a) esquema de extração de cotas (b) cotas extraídas para a vista exibida em (a). 100
4.14 Mapeamento do tipo II: (a) esquema de extração de cotas (b) cotas extraídas para a vista exibida em (a). 101
4.15 Mapeamento do tipo III: (a) esquema de extração de cotas (b) cotas extraídas para a vista exibida em (a). 101
4.16 Mapeamento do tipo IV: (a) esquema de extração de cotas (b) cotas extraídas para a vista exibida em (a). 103
4.17 Mapeamento do tipo V: (a) esquema de extração de cotas (b) cotas extraídas para a vista exibida em (a). 103
4.18 Diagrama de blocos do sistema de detecção e classificação de vistas de navios baseado em curvas principais. 104
4.19 Histogramas de erros para o classificador I, para dados de teste, em função da classe (cl) e dos ângulos de azimute (az) ou elevação (el). Da esquerda para a direita e de cima para baixo: (a) cl A, az; (b) cl A, el; (c) cl B, az; (d) cl B, el; (e) cl C, az; (f) cl C, el; (g) cl D, az; (h) cl D, el; (i) cl E, az; (j) cl E, el. 109

4.20 Histogramas de erros para o classificador II, para dados de teste, em função da classe (cl) e dos ângulos de azimute (az) ou elevação (el). Da esquerda para a direita e de cima para baixo: (a) cl A, az; (b) cl A, el; (c) cl B, az; (d) cl B, el; (e) cl C, az; (f) cl C, el; (g) cl D, az; (h) cl D, el; (i) cl E, az; (j) cl E, el. 112
4.21 Histogramas de erros para o classificador III, para dados de teste, em função da classe (cl) e dos ângulos de azimute (az) ou elevação (el). Da esquerda para a direita e de cima para baixo: (a) cl A, az; (b) cl A, el; (c) cl B, az; (d) cl B, el; (e) cl C, az; (f) cl C, el; (g) cl D, az; (h) cl D, el; (i) cl E, az; (j) cl E, el. 115
4.22 Histogramas de erros para o classificador IV, para dados de teste, em função da classe (cl) e dos ângulos de azimute (az) ou elevação (el). Da esquerda para a direita e de cima para baixo: (a) cl A, az; (b) cl A, el; (c) cl B, az; (d) cl B, el; (e) cl C, az; (f) cl C, el; (g) cl D, az; (h) cl D, el; (i) cl E, az; (j) cl E, el. 117
4.23 Histogramas de erros para o classificador V, para dados de teste, em função da classe (cl) e dos ângulos de azimute (az) ou elevação (el). Da esquerda para a direita e de cima para baixo: (a) cl A, az; (b) cl A, el; (c) cl B, az; (d) cl B, el; (e) cl C, az; (f) cl C, el; (g) cl D, az; (h) cl D, el; (i) cl E, az; (j) cl E, el. 120
4.24 Dados sem ruído: (a) vista arbitrária da classe C e (b) cotas extraídas dessa vista segundo o mapeamento do tipo 1. 122
4.25 Resultado da adição de ruído gaussiano com valores crescentes de variância às cotas mostradas no gráfico da Figura 4.24(b): (a) $\sigma^2 = 0,005$; (b) $\sigma^2 = 0,01$; (c) $\sigma^2 = 0,02$; (d) $\sigma^2 = 0,05$; (e) $\sigma^2 = 0,1$; (f) $\sigma^2 = 0,2$. 122
4.26 Eficiência média dos classificadores Tipo I, III, IV e V para dados de teste aos quais se adicionou ruído gaussiano com valores de variância na faixa compreendida entre 0,005 e 0,2. 123

Lista de Tabelas

2.1 Comparação dos principais algoritmos de obtenção de curvas principais. As siglas HS, BR, PLA, PCOP e k-seg se referem, respectivamente, aos algoritmos propostos por Hastie e Stuetzle, Banfield e Raftery, Kégl et al., Delicado e Verbeek et al.(k-segmentos não-suave). 39

3.1 Quantidade de espectros que compõem os conjuntos de dados de cada classe. 52

3.2 Quantidade de espectros que compõem os conjuntos de projeto e de teste de cada classe. 55

3.3 Índices dos dez tons de freqüência de maior intensidade, na média, irradiados por navios das diversas classes. 56

3.4 Distâncias euclidianas entre os centros das classes (u.a.). 61

3.5 Média das distâncias euclidianas entre espectros (u.a.). 62

3.6 Eficiências de detecção das classes em função do número de segmentos e do pré-processamento adotado. Resultados obtidos para dados de projeto em porcentagem. 67

3.7 Número máximo de segmentos e comprimentos das curvas de cada classe (u.a.), empregando a normalização AN2. 70

3.8 Número máximo de segmentos e comprimentos das curvas de cada classe (u.a.), empregando a normalização AN3. 70

3.9 Média das distâncias entre espectros e curvas (u.a.). 71

3.10 Distâncias mínimas entre curvas (u.a.). 73

3.11 Distâncias de *Hausdorff* entre curvas (u.a.). 74

3.12 Média ponderada das distâncias mínimas entre curvas (u.a.). 75

3.13 Matriz de confusão para o melhor classificador obtido pela metodologia AN2. Resultados para dados de *projeto* em porcentagem. 76

3.14 Matriz de confusão para o melhor classificador obtido pela metodologia AN2. Resultados para dados de *teste* em porcentagem. 77
3.15 Matriz de confusão para o melhor classificador obtido pela metodologia AN3. Resultados para dados de *projeto* em porcentagem. 77
3.16 Matriz de confusão para o melhor classificador obtido pela metodologia AN3. Resultados para dados de *teste* em porcentagem. 78
3.17 Comparação da eficiência de classificação de oito classes de navios empregando-se diversos métodos. 81

4.1 Quantidade de vértices e planos empregados na construção dos modelos das cinco classes de navios. 96
4.2 Número de segmentos que compõem as curvas principais das diversas classe extraídas a partir das cotas obtidas empregando-se os cinco métodos de mapeamento. 106
4.3 Matriz de confusão para o classificador I. Resultados para dados de *projeto* em porcentagem. 108
4.4 Matriz de confusão para o classificador I. Resultados para dados de *teste* em porcentagem. 108
4.5 Matriz de confusão para o classificador II. Resultados para dados de *projeto* em porcentagem. 111
4.6 Matriz de confusão para o classificador II. Resultados para dados de *teste* em porcentagem. 111
4.7 Matriz de confusão para o classificador III. Resultados para dados de *projeto* em porcentagem. 113
4.8 Matriz de confusão para o classificador III. Resultados para dados de *teste* em porcentagem. 114
4.9 Matriz de confusão para o classificador IV. Resultados para dados de *projeto* em porcentagem. 116
4.10 Matriz de confusão para o classificador IV. Resultados para dados de *teste* em porcentagem. 116
4.11 Matriz de confusão para o classificador V. Resultados para dados de *projeto* em porcentagem. 119

4.12 Matriz de confusão para o classificador V. Resultados para dados de *teste* em porcentagem. 119
4.13 Comparação do desempenho de cada tipo de classificador para dados de teste ruidosos. Valores de variância do ruído (σ_n^2) em unidades arbitrárias (u.a.) e de eficiência média de classificação em porcentual (%). 123
4.14 Comparação do desempenho de cada tipo de classificador para dados de projeto e teste. Valores de eficiência média de classificação em porcentual (%). 124

Capítulo 1

Introdução

Problemas freqüentes em várias áreas de estudo, tais como a engenharia, a estatística, a economia e a medicina, empregam a representação de dados em espaços de dimensão elevada. A representação compacta desses dados e a extração de informação relevante são áreas de grande interesse. Frequentemente, observa-se que é possível compactar dados, com perda irrelevante ou aceitável na informação. Neste caso, é possível reduzir significativamente o tempo de processamento, possibilitando, inclusive, o processamento em tempo real de tarefas complexas. A compactação dos dados também permite que se eleve a eficiência dos sistemas e se reduzam os requisitos de hardware envolvidos, tais como a capacidade de armazenamento de dados. Várias técnicas foram propostas visando a compactação dos dados, sendo, provavelmente, a técnica de análise de componentes principais (PCA) [3] uma das mais conhecidas e das mais frequentemente utilizadas. Outras técnicas exploram estatísticas de ordem superior, alcançando excelentes resultados de compactação. Dentre elas, podemos citar as técnicas de análise de componentes não-lineares (NLPCA) [4] , análise de componentes de discriminação (PCD) [3], análise de componentes independentes (ICA) [5] e curvas principais [1], a técnica que será explorada durante o desenvolvimento deste trabalho.

A extração de informação a partir de dados experimentais é fundamental, visto que, em muitos casos, a modelagem adequada do processo é muito complexa, sendo necessário que se extraia informação relevante a partir de observações do processo, nas suas diversas condições operativas. A técnica de curvas principais, conforme será demonstrado através das duas aplicações apresentadas nos Capítulos

3 e 4, é uma ferramenta poderosa de representação, podendo ser empregada na extração de assinaturas, a partir de dados experimentais.

As aplicações apresentadas tratam de sistemas de apoio à decisão, cuja construção envolve o processamento de um grande volume de dados e emprega técnicas de processamento como a amostragem, a filtragem, a normalização, a extração de características e a classificação. Esses sistemas são, geralmente, acoplados a equipamentos de vigilância empregados a bordo de navios, tais como sonares e câmeras infravermelhas, os quais coletam uma grande quantidade de dados, que, necessariamente, devem ser processados rapidamente, de forma a permitir que os operadores os analisem e tomem decisões que são fundamentais à segurança dos meios. Os sistemas de apoio à decisão aumentam a confiabilidade das escolhas tomadas pelos operadores, sobre os quais recai a difícil tarefa de decidir quando alertar sobre a presença de uma ameaça ou de uma situação de interesse.

1.1 Motivação

A vigilância é, possivelmente, a tarefa mais corriqueira e das mais essenciais dentre as realizadas pelas Forças Armadas. Diversos são os equipamentos empregados, cobrindo faixas de freqüência que vão desde as mais baixas até a região do infravermelho. O objetivo desses equipamentos é o de monitorar o ambiente em busca de alvos, sejam eles ameaças ou objetos de interesse. Pouca seria a utilidade desses equipamentos se as informações fornecidas por eles não pudessem ser interpretadas convenientemente por seus operadores. Por exemplo, um radar de vigilância instalado a bordo de um navio monitora o espaço ao redor da plataforma em busca de alvos de interesse. Se uma aeronave aproxima-se do navio é necessário que o radar indique, permanentemente, sua posição e sua seção reta[1]. Com base na seqüência de informações fornecidas pelo radar, o operador, além de conhecer a posição do alvo, pode obter sua velocidade e estimar suas dimensões e, assim, concluir que se trata de uma aeronave e iniciar o processo de identificação do contato. A classificação de alvos é uma tarefa fundamental, que deve ser executada em tempo hábil, permitindo que se tomem as ações de contramedidas necessárias. Além disso, busca-se

[1] Área do alvo estimada a partir da intensidade do sinal refletido por ele.

atingir um grau de confiabilidade elevado, dotando os equipamentos de vigilância de classificadores automáticos que auxiliem os operadores na tomada de decisão.

Para um submarino de guerra, a identificação da classe de um navio de superfície ou até de outro submarino, a partir de sinais acústicos captados pelo seu sonar passivo, é de grande importância para o seu desempenho em ações de patrulha e ataque. A fim de desempenhar esta tarefa, além de ouvir os sinais de áudio captados pelos transdutores, os operadores de sonar consideram os resultados de diversas análises produzidas pelo equipamento. Ainda assim, a identificação de uma classe de navio não é uma tarefa simples, mesmo para operadores muito experientes.

Diversos tipos de armamento orientam-se em direção ao alvo guiados pelo calor emitido por ele ou por caminhos que impedem sua detecção por sensores que monitoram a faixa de freqüência de radar, empregada por diversos tipos de equipamentos de defesa encontrados em operação a bordo de navios e aeronaves. Essas ameaças, no entanto, podem ser detectadas pela análise de imagens infravermelhas, obtidas a partir de sensores específicos.

É fundamental que um classificador, para emprego naval, possua as seguintes características:

- Alta eficiência média de classificação.
- Robustez.
- Possa operar em tempo real ou possua um tempo de reação que permita a tomada de ações de contramedidas adequadas.

Além disso, é desejável que ele possua as seguintes características adicionais:

- Permita que o usuário final possa realizar, de forma trivial, as tarefas de inserção e de remoção de novas classes.
- Empregue um banco de dados composto de assinaturas das classes.

Como será visto nos capítulos subsequentes, os classificadores baseados em curvas principais possuem as características apontadas, sendo uma opção atraente aos classificadores com arquiteturas mais tradicionais.

1.2 O que foi feito

As seguintes tarefas abaixo listadas se referem ao desenvolvimento do classificador de navios baseado em sinais de sonar passivo:

- Apresentação do conjunto de dados e do esquema de pré-processamento dos sinais de sonar passivo, desenvolvido por Soares-Filho [6], a fim de extrair os tons de freqüência que serão empregados na classificação das classes de navios.
- Discussão sobre a viabilidade da classificação de navios, baseada em tons de freqüência irradiados durante o deslocamento de uma navio e uma breve análise estatística desses tons, incluindo a apresentação de ajustes de funções de densidade de probabilidade de tons de freqüência selecionados.
- Análise espacial da distribuição dos espectros.
- Introdução de uma técnica de normalização e o emprêgo de médias de espectros.
- Forma de extração de curvas principais, empregando o algoritmo de k-segmentos não-suave, desenvolvido por J. J. Verbeek [7], e análise da distribuição espacial das curvas e entre curvas e espectros.
- Apresentação da implementação do classificador; apresentação e discussão dos resultados de classificação à luz das análises realizadas; apresentação da implementação do classificador em linguagem C; comparação dos resultados alcançados com os obtidos por outros autores, em trabalhos que empregaram os mesmos conjuntos de dados; e análise preliminar da robustez do classificador.

As demais tarefas abaixo listadas se referem ao desenvolvimento do classificador de navios baseado em imagens infravermelhas segmentadas:

- Apresentação do conjunto de dados.
- Apresentação dos métodos propostos para a extração de características discriminantes.
- Implementação do classificador.

- Apresentação e análise dos resultados de classificação.
- Análise preliminar da robustez do classificador baseada em dados ruidosos.

1.3 Organização do trabalho

No capítulo 2 é decrita a técnica de análise de curvas principais. Desde o trabalho pioneiro, publicado por Hastie e Stuetzle em 1989, diversas foram as contribuições que resultaram em algoritmos eficientes, os quais permitem o emprego desta técnica em aplicações de tempo real, como as que serão propostas nos capítulos seguintes. Neste capítulo são abordados os principais trabalhos publicados na literatura, cobrindo as definições propostas, os algoritmos de extração de curvas principais e as aplicações apresentadas. Ênfase especial é dada ao algoritmo de k-segmentos não-suave, empregado na extração das curvas principais usadas na aplicações abordadas nos capítulos subsequentes. Ao fim do capítulo, os principais algoritmos de extração de curvas principais são comparados sob diversos aspectos práticos.

Iniciando a parte prática deste trabalho, no capítulo 3 é proposto um classificador para oito classes distintas de navios, a partir de sinais de sonar passivo. Inicialmente, são abordados trabalhos similares, onde técnicas diversas são empregadas com o objetivo de classificar alvos a partir dos ruídos emitidos por eles. A seguir, são descritos os dados e apresentada a cadeia de pré-processamento. Posteriormente, uma breve análise da distribuição estatística dos sinais empregados é apresentada, a qual é seguida por detalhes de implementação do classificador. Antes da apresentação dos resultados de eficiência de classificação, os quais são comparados com os resultados obtidos em outros trabalhos, são apresentadas medidas de distância entre curvas, as quais visam à análise dos resultados de classificação obtidos. Finalmente, é apresentado um resumo dos assuntos tratados no capítulo.

Dando prosseguimento às aplicações propostas, o capítulo 4 trata da aplicação da técnica de curvas principais à classificação de cinco classes distintas de navios, com base em imagens infravermelhas segmentadas. Inicialmente, são apresentados os conjuntos de dados, os quais são obtidos a partir de modelos tridimensionais. Em seguida, são apresentados os métodos de parametrização propostos, as eficiências de classificação obtidas e a distribuições dos erros em função dos ângulos de tomada das

imagens. A seguir, os resultados obtidos são comparados com os apresentados por outro autor, o qual propõe um classificador neural baseado em momentos invariantes. Um sumário dos assuntos tratados encerra o capítulo.

Por fim, o capítulo 5 apresenta as conclusões obtidas ao final da elaboração deste trabalho, apresentando suas contribuições, e finalizando com os possíveis trabalhos futuros.

Capítulo 2

Curvas Principais

As curvas principais consistem numa generalização não-linear da técnica de análise de componentes principais. Elas foram introduzidas por Hastie [8] e Hastie e Stuetzle [1], doravante designados HS, como curvas suaves, unidimensionais, que *passam no meio* do conjunto multidimensional de dados, fornecendo um bom resumo unidimensional destes. Além disso, elas são não-paramétricas e sua forma é sugerida pelos dados.

Desde o trabalho inicial de HS, as curvas principais despertaram bastante interesse da comunidade científica, tendo sido publicados diversos trabalhos que aprimoram a definição inicial, apresentam definições alternativas e propõem aplicações.

Neste capítulo serão abordados os principais trabalhos publicados sobre curvas principais. Na seção 2.1, o artigo de HS será apresentado. Serão discutidas as definições e os algoritmos propostos pelos autores para a obtenção de curvas principais. Também será discutida a tendência (*bias*) das curvas principais obtidas com o emprego do algoritmo proposto por HS e como um dos termos da tendência foi minimizado por Banfield e Raftery [2]. A seguir, na seção 2.2, serão apresentadas definições alternativas às propostas por HS. Ainda nesta seção, será detalhado o algoritmo de k-segmentos não-suave [7] desenvolvido por Verbeek, Vlassis e Kröse, o qual foi empregado para a extração das curvas principais necessárias às aplicações que serão propostas neste trabalho. Na seção 2.3 serão apresentadas algumas aplicações publicadas na literatura. Por fim, os tópicos abordados neste capítulo serão resumidos na seção 2.4.

2.1 Curvas principais auto-consistentes

A fim de ilustrar a noção de representação de um conjunto de dados empregando uma curva principal, HS apresentam um diagrama de dispersão o qual é reproduzido na Figura 2.1. Esta figura mostra diversos eventos $\mathbf{x}$ pertencentes a um conjunto de dados em duas dimensões, $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2)$.

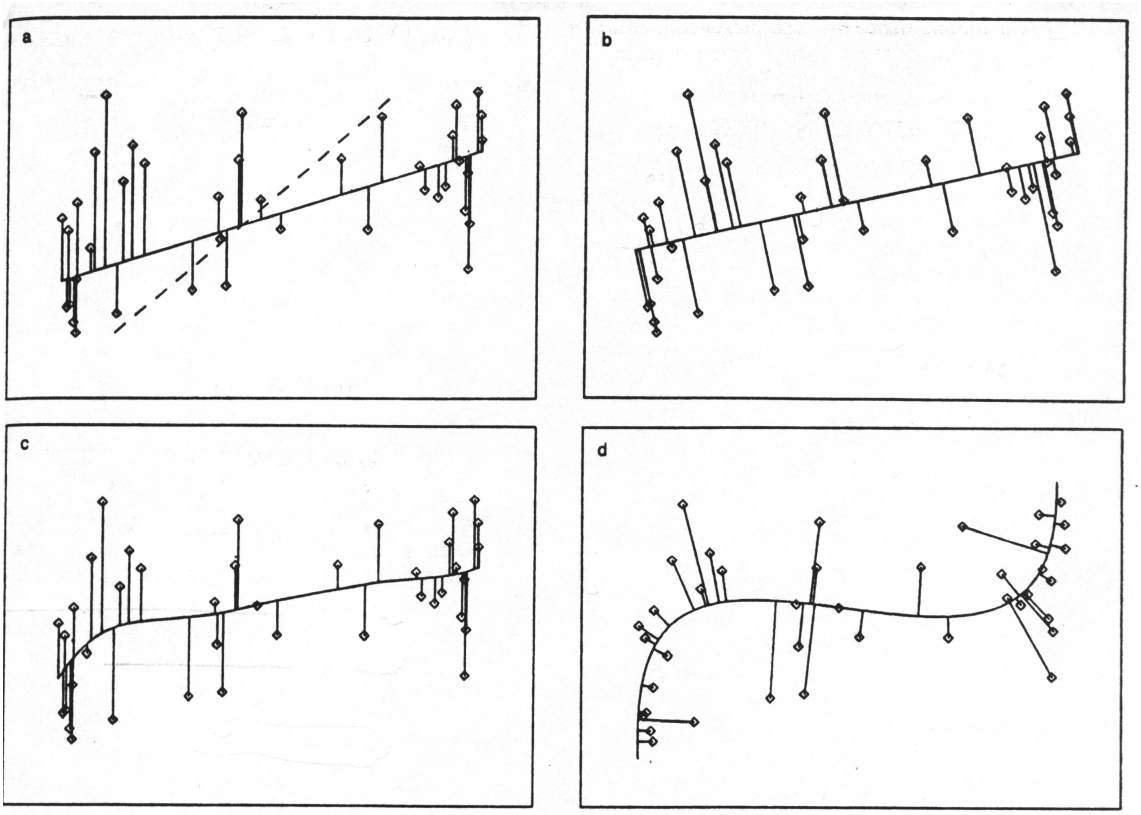


Figura 2.1: Algumas formas de representação de dados: (a) regressão linear; (b) primeira componente principal; (c) regressão não-linear; e (d) curva principal (extraída de [1]).

Se as duas variáveis, $\mathbf{X}_1$ e $\mathbf{X_2}$, são dependentes entre si, tal que $\mathbf{X}_2$ é a resposta a alterações na variável independente, $\mathbf{X}_1$, pode-se buscar uma relação linear entre as duas variáveis por meio de algum método de regressão linear. Neste caso, $\mathbf{X}_2$ é modelado como uma função linear de $\mathbf{X}_1$, obtendo-se a direção que minimiza o quadrado das distâncias verticais entre cada evento $\mathbf{x}$ e a linha reta, conforme mostrado na Figura 2.1 (a).

Se as duas variáveis devem ser tratadas de forma simétrica, ou seja, se as variáveis são independentes, a primeira componente principal é a direção que minimiza as distâncias ortogonais dos pontos à reta, conforme mostrado na Figura 2.1

(b).

Além da regressão linear, é possível empregar funções não-lineares a fim de minimizar a distância vertical entre a variável dependente, $\mathbf{X}_2$, e a independente, $\mathbf{X}_1$. A Figura 2.1(c) mostra um exemplo de curva obtida através de regressão não linear dos dados.

Finalmente, na Figura 2.1(d) é mostrada uma curva principal obtida a partir dos dados. Vimos que a primeira componente principal é a melhor aproximação linear que minimiza a distância ortogonal entre a reta e os dados. No caso da curva principal, por ser uma aproximação não-linear, cada um dos seus pontos tende a minimizar a distância ortogonal entre os dados e a curva.

Muito embora a apresentação de curvas principais como *curvas que passam no meio da nuvem de dados*, seja bastante intuitiva, o conceito de *passar no meio dos dados* é abstrato, devendo ser definido de forma precisa. Formalmente, HS apresentam a definição de curvas principais a partir de uma distribuição de probabilidade e, em seguida, esses conceitos são estendidos para o caso prático, onde, normalmente, a função de densidade de probabilidade é desconhecida, mas dispõe-se de um conjunto de dados consistindo de n realizações do fenômeno que se deseja representar.

2.1.1 Definição de HS

Conforme será detalhado a seguir, HS definem curvas principais a partir do conceito de auto-consistência, o qual, por sua vez, é definido com base no índice de projeção de um ponto $\mathbf{x}$ numa curva $\mathbf{f}$.

Uma curva unidimensional, num espaço de d dimensões, $\Re^d$, é um vetor $\mathbf{f}(t)$ de d funções contínuas de uma única variável t, ou seja, $\mathbf{f}(t) = \{\mathbf{f}_1(t), \ldots, \mathbf{f}_d(t)\}$. Essas funções são denominadas *funções de coordenada* e o parâmetro t provê ordenamento ao longo da curva.

Seja $\mathbf{X}$ um vetor aleatório em $\Re^d$, possuindo densidade de probabilidade h, e momentos de segunda ordem finitos. Seja $\mathbf{f}$ uma curva suave, de velocidade unitária[1] em $\Re^d$, parametrizada no intervalo fechado $I \subseteq \Re$, que não intercepta a si própria (ou seja, $t_1 \neq t_2 \Rightarrow \mathbf{f}(t_1) \neq \mathbf{f}(t_2)$), e de comprimento finito dentro de uma esfera de

[1] O comprimento l de uma curva $\mathbf{f}(t)$, de t_0 a t_1, é obtido pela seguinte equação: $l = \int_{t_0}^{t_1} \|\mathbf{f}'(z)dz\|$. Se $\mathbf{f}'(z) = 1$, então, $l = \| t_1 - t_0 \|$ e $\mathbf{f}(t)$ é uma curva de velocidade unitária.

dimensões finitas em $\Re^d$.

O *índice de projeção* $t_{\mathbf{f}} : \Re^d \rightarrow \Re$ é definido como

$$t_{\mathbf{f}}(\mathbf{x}) = sup_t\{t : \|\mathbf{x} - \mathbf{f}(t)\| = inf_\mu\|\mathbf{x} - \mathbf{f}(\mu)\|\}, \tag{2.1}$$

onde $\mathbf{x}$ é um evento arbitrário pertencente a $\mathbf{X}$ e μ é uma variável auxiliar definida em $\Re$. O índice de projeção $t_{\mathbf{f}}(\mathbf{x})$ é o valor de t para o qual a curva principal $\mathbf{f}(t)$ está mais próxima de $\mathbf{x}$. Se houver mais de um valor possível, o maior deles é selecionado. A Figura 2.2 ilustra o conceito de índice de projeção relativo a uma curva principal. São mostrados cinco eventos $\mathbf{x}_1, \cdots, \mathbf{x}_5$, os quais projetam na curva, respectivamente, nos pontos $\mathbf{f}(t_f(\mathbf{x}_1)), \cdots, \mathbf{f}(t_f(\mathbf{x}_5))$. Observe que $t = t_f(\mathbf{x}_i)$ é o valor do parâmetro t relativo ao ponto da curva mais próximo do evento $\mathbf{x}_i$ e, por isso, corresponde ao valor do índice de projeção desse ponto sobre a curva, $t_f(\mathbf{x}_i)$.

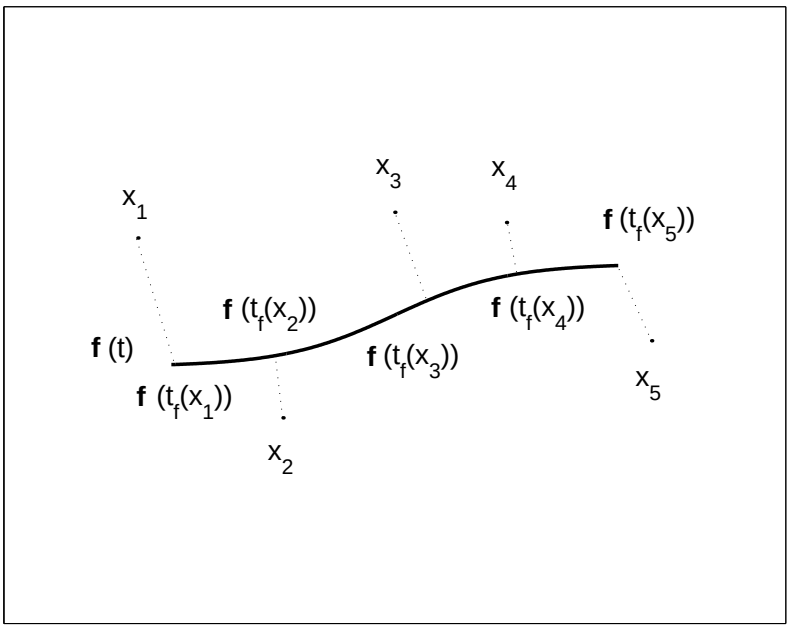


Figura 2.2: Modelo de uma curva principal.

Conforme definido por HS, uma curva principal $\mathbf{f}(t)$, definida em $\Re^d$, possui as seguintes características:

1. É uma curva suave (infinitamente diferenciável);

2. Não intercepta a si própria ($t_1 \neq t_2 \Rightarrow \mathbf{f}(t_1) \neq \mathbf{f}(t_2)$);

3. Possui um comprimento finito dentro de um espaço finito em $\Re^d$;

4. É *auto-consistente*.

Esta última característica decorre da propriedade dos pontos que compõem uma curva principal consistirem na média dos dados que nela projetam. Uma curva é *auto-consistente* ou uma *curva principal* da distribuição h se, para todos os valores do parâmetro t, seus pontos forem a média das realizações de $\mathbf{X}$ que nela projetam ortogonalmente, ou seja, se

$$\mathbf{f}(t) = \mathbf{E}[\mathbf{X} \mid t_{\mathbf{f}}(\mathbf{X}) = t], \qquad \forall\ t. \tag{2.2}$$

Note que o índice de projeção foi definido para um evento arbitrário, conforme a Equação 2.1, no entanto, a definição apresentada na Equação 2.2, emprega não o evento arbitrário $\mathbf{x}$, mas o vetor aleatório $\mathbf{X}$. Neste caso, $t_{\mathbf{f}}(\mathbf{X})$ representa todos os valores que a variável t pode assumir, dado um vetor aleatório $\mathbf{X}$, ou seja, $t_{\mathbf{f}}(\mathbf{X}) = t$.

A Figura 2.3 ilustra um modelo de curva principal para uma distribuição de dados arbitrária em duas dimensões. Nela são mostrados apenas oito eventos que determinam o ponto $\mathbf{f}(t_i)$ da curva principal. Idealmente, no entanto, existe uma quantidade ilimitada de eventos, sendo cada um dos pontos da curva principal definido pela média dos eventos que projetam exatamente nele.

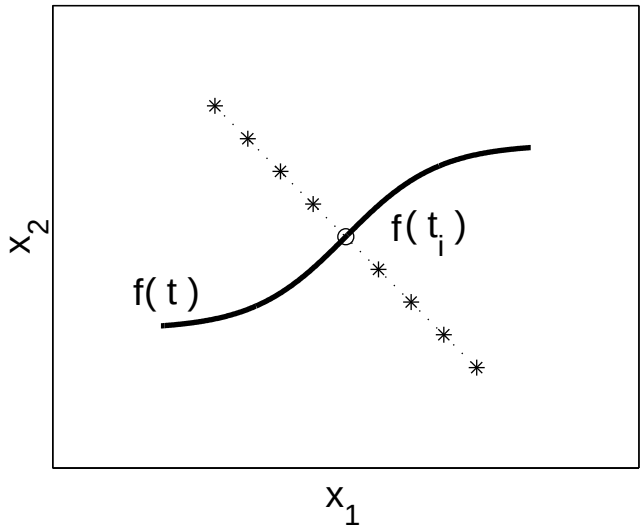


Figura 2.3: Curva principal obtida para uma distribuição de dados.

Baseado nas Equações 2.1 e 2.2, HS propõem um algoritmo iterativo de ex-

tração de curvas principais [2], conforme será apresentado no item 2.1.2.

Neste ponto, é necessário que se apresentem as definições de função de distância e de função de distância empírica [9], visto que estes conceitos serão empregados ao longo de todo este trabalho. A seguir, o conceito de auto-consistência será revisto a partir de duas das proposições apresentadas por HS [1].

Uma curva definida num espaço euclidiano é uma função contínua $\mathbf{f} : \mathbf{I} \rightarrow \Re^d$, onde $\mathbf{I}$ é um intervalo fechado em $\Re$. A *função de distância* entre $\mathbf{X}$, de distribuição h, e a curva $\mathbf{f}$ é definida como o valor esperado do quadrado da distância euclidiana entre $\mathbf{X}$ e $\mathbf{f}$, ou seja,

$$\Delta(h, \mathbf{f}) = \mathbf{E}\,[\, inf_t \,\|\mathbf{X} - \mathbf{f}(t)\|^2\,] = \mathbf{E}\,\|\,\mathbf{X} - \mathbf{f}(t_{\mathbf{f}}(\mathbf{X}))\,\|^2, \tag{2.3}$$

onde o índice de projeção $t_{\mathbf{f}}(\mathbf{x})$ é dado pela Equação 2.1.

Caso a distribuição de $\mathbf{X}$ não seja conhecida mas, no entanto, se disponha de n realizações independentes e identicamente distribuídas (i.i.d.), ao invés da função de distância emprega-se a *função de distância empírica*, definida como a média das distâncias ortogonais de cada evento pertencente ao conjunto de dados à curva, ou seja,

$$\Delta_n(\mathbf{f}) = \frac{1}{n}\sum_{i=1}^{n} \Delta(\mathbf{x}_i, \mathbf{f}), \tag{2.4}$$

onde $\Delta(\mathbf{x}_i, \mathbf{f})$ é a distância entre o evento $\mathbf{x}_i$ à curva $\mathbf{f}$.

Se $\mathbf{f}(t)$ é uma curva poligonal, a distância de um evento $\mathbf{x}$ à curva depende do valor do índice de projeção $t_{\mathbf{f}}(\mathbf{x})$. A Figura 2.4 mostra dois eventos $\mathbf{x}_1$ e $\mathbf{x}_2$, definidos em $\Re^2$ e representados por dois pontos no plano cartesiano, e uma curva poligonal, também definida no plano, a qual possui apenas dois segmentos, $\mathbf{s}_1$ e $\mathbf{s}_2$,

[2]Note que a Equação 2.1 define o índice de projeção de um evento dada uma curva principal, enquanto que a Equação 2.2 define a curva principal dado que se dispõem dos valores dos índices de projeção dos eventos sobre a curva, sugerindo o emprego de um procedimento iterativo na solução do problema de extração de uma curva principal. De fato, o algoritmo proposto por HS é iterativo: partindo-se de uma primeira aproximação para a curva principal procurada, inicialmente, projetam-se os eventos sobre esta curva e, em seguida, obtêm-se o somatório das distâncias entre esses pontos e a curva. A seguir, obtêm-se uma segunda aproximação da curva e retorna-se ao primeiro passo, buscando sempre minimizar o valor médio do quadrado das distâncias entre os eventos e a curva.

sendo o primeiro definido pelos vértices $\mathbf{v}_1$ e $\mathbf{v}_2$ e o segundo por $\mathbf{v}_2$ e $\mathbf{v}_3$. Nota-se que os dois eventos estão mais próximos do primeiro segmento, portanto, é a partir dele que devem ser obtidas as distâncias. O evento $\mathbf{x}_1$ projeta, ortogonalmente, sobre o primeiro segmento. Neste caso, a distância é medida do ponto de projeção ao evento em questão. O evento $\mathbf{x}_2$, no entanto, não projeta sobre o segmento, sendo o vértice $\mathbf{v}_1$ o ponto pertencente ao segmento que está mais próximo dele. Neste caso, a distância será tomada entre o evento $\mathbf{x}_2$ e o vértice $\mathbf{v}_1$ [9].

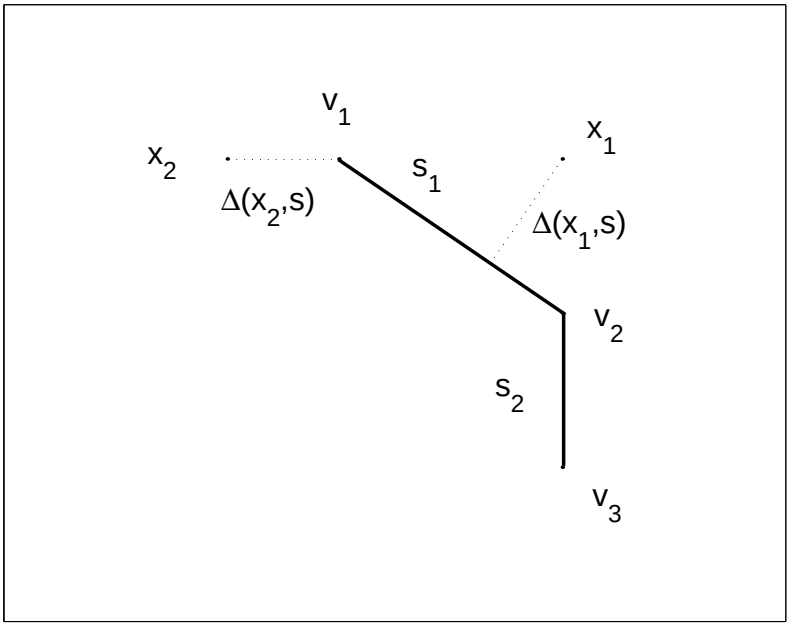


Figura 2.4: Distância de um ponto a uma curva poligonal.

Considere uma reta definida pela equação $\mathbf{l}(t) = t\mathbf{u} + \mathbf{c}$ onde $\mathbf{u}, \mathbf{c} \in \Re^d$, $\mathbf{u}$ é o vetor unitário que define a direção da reta e $\mathbf{v}_1$ e $\mathbf{v}_2$ são pontos pertencentes à reta e vértices do segmento de reta $\mathbf{s}(t)$. A distância de um evento $\mathbf{x} \in \Re^d$ ao segmento $\mathbf{s}(t)$ é definida como o quadrado da distância euclidiana entre o evento e seu ponto de projeção sobre a curva, o qual, conforme mostrado na Figura 2.4, pode ser um dos pontos pertencentes à curva ou um dos vértices do segmento. Matematicamente, a distância do evento $\mathbf{x}$ ao segmento $\mathbf{s}$ é definido como

$$\Delta(\mathbf{x}, \mathbf{s}) = \begin{cases} \|\mathbf{x} - \mathbf{v}_1\|^2 & \text{se } \mathbf{s}(t_s(\mathbf{x})) = \mathbf{v}_1, \\ \|\mathbf{x} - \mathbf{v}_2\|^2 & \text{se } \mathbf{s}(t_s(\mathbf{x})) = \mathbf{v}_2, \\ \|\mathbf{x} - \mathbf{c}\|^2 - ((\mathbf{x} - \mathbf{c})^T\mathbf{u})^2 & \text{qualquer outro caso,} \end{cases} \tag{2.5}$$

onde $\|\mathbf{y}\| = [abs(y_1)^2 + \cdots + abs(y_d)^2]^{1/2}$ para $\mathbf{y} \in \Re^d$, a função $abs(n)$ retorna o valor absoluto de n para $n \in \Re$ e T representa a operação de transposição.

Antes de considerar uma curva principal para conjuntos de dados, HS estabelecem e provam algumas proposições que comprovam que curvas principais podem ser consideradas como uma generalização do conceito de componentes principais [1].

Proposição 1: Se uma linha reta[3] $\mathbf{l}(t) = \mathbf{u} + t\mathbf{v}$ é auto-consistente, então, $\mathbf{l}(t)$ é uma componente principal. Ainda segundo HS, uma componente principal não é necessariamente auto-consistente, no entanto, ela é auto-consistente com relação à regressão linear, ou seja, uma componente principal (especificamente, a primeira componente principal) possui a direção que minimiza a função de distância dentre todas as possíveis direções de projeção.

Proposição 2: Sabe-se que as componentes principais são pontos críticos da função de distância. Especificamente, a primeira componente principal é a direção que minimiza a função de distância dentre todas as possíveis direções. Considere um vetor aleatório $\mathbf{X}$, de densidade de probabilidade h, e uma reta $\mathbf{l}(t) = \mathbf{u} + t\mathbf{v}$. A função de distância dos eventos gerados a partir da distribuição h à reta $\mathbf{l}$, $\Delta(h, \mathbf{l})$, pode ser representada em função do ponto $\mathbf{u}$ e do vetor unitário $\mathbf{v}$, $\Delta(h, \mathbf{l}) = \Delta(h, \mathbf{u}, \mathbf{v})$. Assim, $grad_{u,v}\Delta(h, \mathbf{u}, \mathbf{v}) = 0$ se, e somente se, $\mathbf{u} = 0$ e $\mathbf{v}$ é um autovetor da matriz de covariância de $\mathbf{X}$, ou seja, se $\mathbf{l}$ for uma componente principal.

Analogamente, considere que $\mathbf{f}_t$ é uma versão perturbada de $\mathbf{f}$, tal que $\mathbf{f}_t = \mathbf{f} + t\mathbf{g}$, onde $\mathbf{f}$ e $\mathbf{g}$ são curvas suaves, $\mathbf{g} \in G$, e t é um parâmetro em $\Re$. HS provam que uma curva $\mathbf{f}$ é uma curva principal se, e somente se, ela é um ponto crítico da função de distância para variações na classe de curvas G, ou seja,

$$\frac{\partial \Delta(h, \mathbf{f}_t)}{\partial t}\Big|_{t=0} = 0 \quad \forall \mathbf{g} \in G, \tag{2.6}$$

concluindo que uma curva principal pode ser vista como uma generalização da técnica de componentes principais. As provas destas proposições podem ser en-

[3]Na equação vetorial de uma reta definida em $\Re^d$ [10], $\mathbf{l}(t) = \mathbf{u} + t\mathbf{v}$, $\mathbf{u} \in \Re^d$ é um ponto pertencente à reta, $t \in \Re$ é o parâmetro que provê ordenação ao longo da reta e $\mathbf{v} \in \Re^d$ é o vetor unitário que define a direção da reta.

contradas em [1].

2.1.2 Algoritmo proposto por HS

Com base no conceito de auto-consistência, HS propõem um algoritmo iterativo para obtenção de uma curva principal para dados gerados a partir de uma distribuição de probabilidade. O algoritmo mostrado a seguir consiste, basicamente, num passo de projeção, no qual se determinam os valores dos índices de projeção, e noutro de cálculo do valor esperado, onde se obtém, a partir dos índices de projeção, os pontos da curva principal na j-ésima iteração. O programa é encerrado quando ocorre a convergência, ou seja, quando o valor esperado da função de distância atinge um valor menor que um limite pré-estabelecido.

Passo 0 - Inicialização: Seja $\mathbf{f}^{(0)}(t) = \overline{\mathbf{x}} + \mathbf{a}t$, onde $\overline{\mathbf{x}}$ é o valor esperado de $\mathbf{X}$ e $\mathbf{a}$ é o vetor unitário que possui a direção da primeira componente principal da distribuição de $\mathbf{X}$, h. Seja $t^{(0)}(\mathbf{x}) = t_{\mathbf{f}^{(0)}}(\mathbf{x}),\ \forall \mathbf{x} \in h$. Seja $j = 1$.

Passo 1 - Estimação: Faça $\mathbf{f}^{(j)}(t) = E(\mathbf{X}|t_{\mathbf{f}^{(j-1)}}(\mathbf{X}) = t)$.

Passo 2 - Projeção: Faça $t^{(j)}(\mathbf{x}) = t_{\mathbf{f}^{(j)}}(\mathbf{x}),\ \forall \mathbf{x} \in h$.

Passo 3 - Convergência: Pare se a diferença entre a função de distância calculada nesta iteração e a obtida na iteração anterior for menor que um valor limite, ou seja, se $|\Delta(h, \mathbf{f}^{(j)}) - \Delta(h, \mathbf{f}^{(j-1)})| < \textit{valor limite}$ onde $\Delta(h, \mathbf{f}^{(j)}) = E\|\mathbf{X} - \mathbf{f}^{(j)}(t_{\mathbf{f}^{(\mathbf{j})}}(\mathbf{X}))\|^2$. Senão, faça $j = j + 1$ e vá para o Passo 1.

Como em casos práticos, usualmente, não se conhece a distribuição dos dados mas, se dispõe de diversas realizações da variável observada, HS também propõem um algoritmo que é capaz de extrair uma curva principal a partir de um conjunto de dados. Neste caso, ao contrário do que ocorre quando se supõe que os dados são gerados a partir de uma distribuição de probabilidade conhecida, apenas se dispõe de uma quantidade limitada de dados. Assim, geralmente, apenas um evento do conjunto de dados $\mathbf{x_i}$ projeta num ponto da curva principal. Nesta situação, não há como definir uma curva principal contínua ou mesmo qualquer um dos seus pontos (já que, por definição, os pontos da curva são definidos como *a média dos eventos* que projetam sobre ela). A fim de resolver este problema, HS propõem que, para

cada evento pertencente ao conjunto de dados $\mathbf{x}_i$, $i = [1, n]$, se obtenha um ponto da curva principal $\mathbf{f}(t_i)$, sendo, cada um destes pontos, deve ser estimado a partir de $k < n$ eventos pertencentes a uma vizinhança de $\mathbf{x}_i$. Conforme discutido em [1], quanto maior o número de eventos empregados na estimação k, mais suave será a curva e menor será a variância de estimação.

Considere então que $\mathbf{X} = (\mathbf{x}_1, \cdots, \mathbf{x}_n)$ é uma matriz $(n \times d)$ de n observações de $\mathbf{X}$ em $\Re^d$. A curva $\mathbf{f}(t)$ é definida a partir de n tuplas (*tuples*) $(t_i, \mathbf{f_i})$, unidas em ordem crescente de t, a fim de formar uma curva poligonal. O parâmetro t assume os valores do comprimento do arco, tal que, $t_1 = 0$ e t_i é igual ao comprimento da curva desde o ponto $\mathbf{f}(t_1)$ até o ponto $\mathbf{f}(t_i)$, definindo uma curva $\mathbf{f}(t)$ que é uma versão discreta de uma curva parametrizada de velocidade unitária. Numa curva poligonal desse tipo, os valores assumidos pelo parâmetro t, t_i podem ser obtidos, recursivamente, da seguinte forma [9]:

$$t_1 = 0, \tag{2.7}$$

$$t_i = t_{i-1} + \|\mathbf{f}(t_i) - \mathbf{f}(t_{i-1})\|, \;\; i = 2, \cdots, n. \tag{2.8}$$

No passo de inicialização, considera-se $\mathbf{f}^{(0)}(t)$ como a linha reta que possui a direção da primeira componente principal dos dados. Ainda durante a inicialização, cada uma das n observações do conjunto de dados é projetada na reta $\mathbf{f}^{(0)}(t)$. Os n pontos de projeção $\mathbf{f}^{(0)}(t_i)$, $\;i = [1, n]$, vão definir o segmento de reta que se constitui na primeira aproximação da curva principal. Um dos pontos de projeção é definido como o ponto correspondente ao primeiro índice de projeção $\mathbf{f}^{(0)}(t = t_1)$ (um dos extremos do segmento de reta). Os demais índices de projeção t_i são obtidos como a distância tomada ao longo da curva entre o ponto $\mathbf{f}(t_1)$ e o ponto de projeção da i-ésima observação $\mathbf{x}_i$, $\mathbf{f}(t_i)$. Como, no Passo de Inicialização, a curva principal é aproximada como uma linha reta, cada índice de projeção t_i é calculado como a distância euclidiana entre $\mathbf{f}(t_1)$ e $\mathbf{f}(t_i)$.

Em seguida, no Passo 1, é aplicado um método de estimação não paramétrica de curvas a fim de unir, de forma suave, cada um dos n pontos, $\mathbf{f}(t_i)$, que vão definir a curva principal obtida nessa iteração. HS propõem o emprego do método LOWRLS[4]

[4]O método LOWRLS empregado por HS se baseia numa regressão linear local ponderada. Em cada ponto, num subconjunto dos dados, uma linha reta é ajustada. O tamanho dos subconjuntos

(*locally weighted running line smoother*), desenvolvido por Cleveland [12], ou de um método baseado em *splines*[5] [13].

No Passo 2 é feita a projeção de cada um dos pontos sobre a curva principal estimada no passo anterior, obtendo-se os n índices de projeção t_i. Esses índices são, então, ordenados e, por fim, é executada uma transformação a fim de obter-se uma curva de velocidade unitária [6].

No Passo 3, a função de distância é substituída pela função de distância empírica (conforme a Equação 2.4). Inicialmente, é obtido o valor da distância empírica, a qual equivale ao somatório dos quadrados das distâncias entre cada ponto $\mathbf{x}_i$ e a curva estimada. Se a diferença entre o valor da função de distância empírica na iteração atual e na anterior é maior que um valor limite definido, o valor de j é incrementado, e o algoritmo volta ao Passo 1, a fim de iniciar uma nova iteração.

A Figura 2.5 mostra uma curva principal obtida a partir de um conjunto finito de n eventos. É mostrado um ponto da curva principal $\mathbf{f}(t_3)$ e a vizinhança de eventos empregada na sua determinação. Observe que $\mathbf{f}(t)$ é definida por exatamente

---

de dados a serem empregados é definido por um coeficiente de suavidade h. Os pontos que compõem o subconjunto são os mais próximos do ponto para o qual se está ajustando a reta. O uso da ponderação se baseia na idéia de que os dados mais próximos do ponto onde se faz o ajuste estão mais correlacionados com ele do que os pontos mais distantes. Como o coeficiente h determina o número de pontos empregados em cada ajuste local, valores grandes de h determinam curvas mais suaves, enquanto que, valores pequenos resultam na interpolação dos dados. Este método foi posteriormente aperfeiçoado por Cleveland e Devlin [11], sendo mais conhecido na literatura como LOWESS (*locally weighted scatterplot smoothing*)

[5]O objetivo do método baseado em *splines* é encontrar uma curva que minimize a função de distância, $\Delta_n(\mathbf{f}) = \frac{1}{n}\sum_{i=1}^{n} \parallel \mathbf{x}_i - \mathbf{f}(t_{\mathbf{f}}(\mathbf{x}_i)) \parallel^2$, e seja suave. HS propõe, então, a minimização da função de distância penalizada

$$G(\mathbf{f}) = \Delta_n(\mathbf{f} + \mu P(\mathbf{f})), \tag{2.9}$$

onde $P(\mathbf{f}) = \int_0^1 \|\mathbf{f}''(\tau)\|^2 d\tau$ mede a curvatura total da curva e μ é um parâmetro de custo cujo valor é mantido no intervalo entre $[0, 1]$.

[6]Sendo uma curva de velocidade unitária, cada índice de projeção arbitrário t_i deve possuir valor igual ao comprimento da curva, tomado desde o ponto extremo da curva, para o qual $t = t_1$, $\mathbf{f}(t_1)$, até o ponto da curva em questão, $\mathbf{f}(t_i)$.

n pontos unidos por meio de segmentos de linha reta.

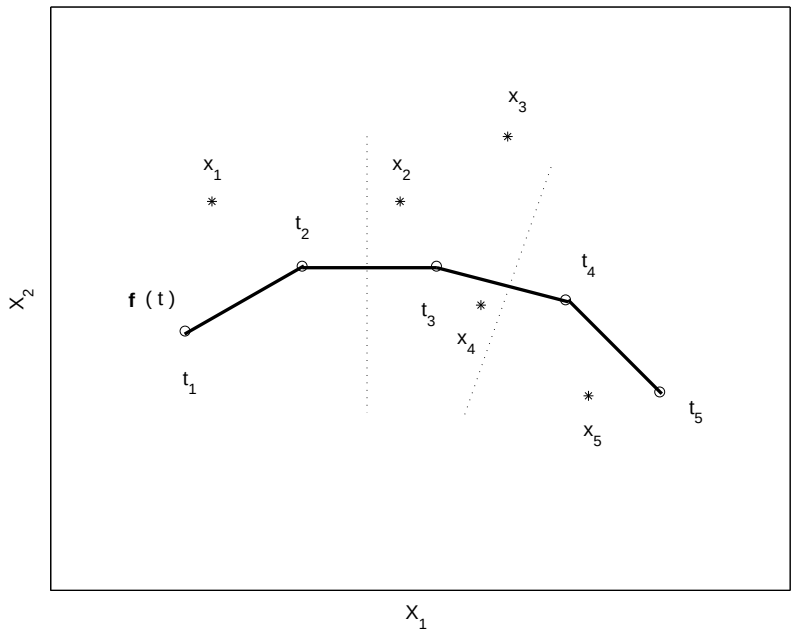


Figura 2.5: Curva principal obtida para um conjunto finito de dados. Cada ponto $\mathbf{f}(t_i)$ da curva é definido como a média ponderada de pontos próximos de $\mathbf{x}_i$ numa vizinhança cujo tamanho é definido por um coeficiente de suavidade.

2.1.3 Problemas potenciais do modelo de HS

HS apontam alguns problemas potenciais [7] no algoritmo proposto. São eles:

1. A curva possui uma tendência (*bias*) em trechos com curvatura.
2. A convergência do algoritmo não é garantida.
3. A existência da curva principal não é garantida para qualquer tipo de distribuição.

A tendência da curva principal obtida através do algoritmo proposto por HS foi, inicialmente, discutida pelos próprios autores [1]. Dois tipos foram observados: a *tendência devida ao modelo* e a *tendência devida à estimação*. O primeiro tipo ocorre quando os dados são gerados a partir do modelo

[7]Designam-se os problemas listados neste item como potenciais porque eles podem ou não ocorrer, conforme será tratado nos parágrafos que se seguem.

$$\mathbf{X} = \mathbf{f}(t) + \mathbf{e} \tag{2.10}$$

onde $\mathbf{e}$ representa o ruído aditivo, o qual é independente de t, e se deseja recuperar $\mathbf{f}(t)$. Em geral, se $\mathbf{f}(t)$ possui uma curvatura, então, ela não é auto-consistente, ou seja, ela não é uma curva principal da distribuição de $\mathbf{X}$. Assim, a curva auto-consistente não coincide com a curva $\mathbf{f}(t)$ estimada, possuindo um raio *maior* com relação ao centro de curvatura.

A tendência devida à estimação ocorre porque os métodos empregados no Passo 1 do algoritmo de HS operam sobre uma vizinhança de pontos. O tamanho dessa vizinhança é definido pelo valor do coeficiente de suavidade, no caso da estimação através do método LOWRLS, ou pelo valor do coeficiente de custo, no caso do emprego de *splines*. Em geral, quanto maior o tamanho da vizinhança, maior será a tendência devida à estimação, acarretando numa curva de raio *menor* que o da curva geratriz. Conforme apontado por HS [1], teoricamente, seria possível cancelar essas duas componentes da tendência. Na maioria das condições práticas, no entanto, a tendência devida à estimação é muito maior que a tendência devida ao modelo [9].

HS apresentam um modelo simples empregado no estudo quantitativo das duas componentes da tendência da curva[8]. Seja $\mathbf{f}(t)$ um círculo parametrizado de curvatura constante $\frac{1}{r}$, ou seja,

$$\mathbf{f}(t) = [(r\ cos(t/r)), (r\ sen(t/r))] \tag{2.11}$$

para $t \in I = [-r\pi, r\pi)$. Considere que os dados $\mathbf{X}$ são gerados conforme o modelo aditivo da Equação 2.10 onde o ruído e é representado por uma gaussiana esférica de média zero e variância igual a σ^2 nas duas dimensões. HS provam que, nesta situação, o raio de curvatura do círculo auto-consistente r^* é maior que o raio da curva geratriz r, conforme mostrado na Figura 2.6. Nesta Figura é mostrado um ponto arbitrário pertencente à curva geratriz $\mathbf{f}(t)$, a qual se deseja obter. O círculo

[8]Uma abordagem mais detalhada é apresentada em [8]. O problema, no entanto, é apresentado de forma clara e concisa em [9].

preenchido, com centro nesse ponto, representa a nuvem de eventos gerados conforme o modelo aditivo. A curva tracejada é a curva auto-consistente $\mathbf{f}^*(t)$, a qual possui raio r^* maior que o raio da curva geratriz r. A explicação intuitiva é que o modelo aditivo parece gerar uma maior quantidade de pontos na parte externa do círculo do que na parte interna, deslocando a curva *mais para fora*. Matematicamente, pode ser mostrado que [8]

$$r^* \approx r + \frac{\sigma^2}{2r}, \tag{2.12}$$

ou seja, a tendência devida ao modelo é igual a $\dfrac{\sigma^2}{2r}$.

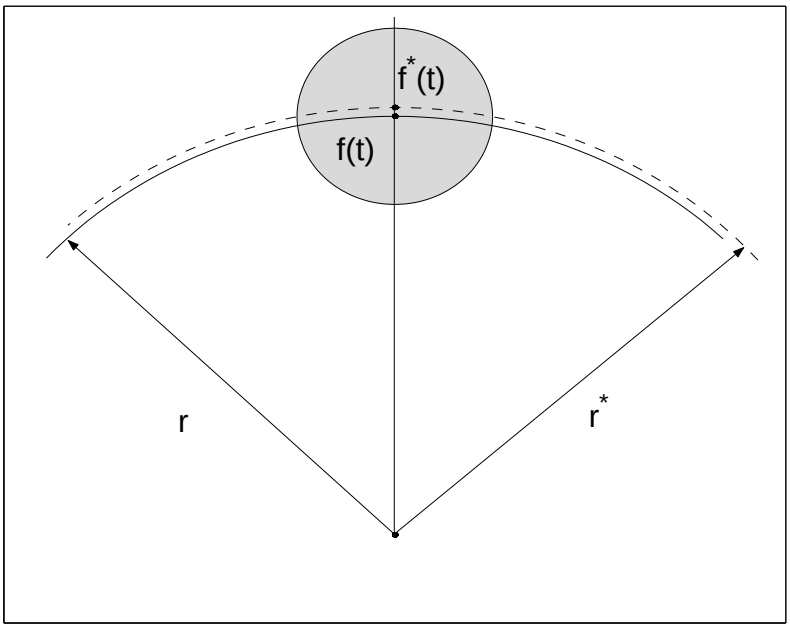


Figura 2.6: Modelo para estudo da tendência devida ao modelo na curva principal definida por HS.

Conforme discutido, a tendência devida à estimação ocorre em virtude do emprego de uma média local. Assuma que a curva principal em $t = t_i$ é estimada empregando-se os pontos que projetam na curva e estão contidos na região entre o arco mais externo e o mais interno, conforme mostrado na Figura 2.7. Considerando um ângulo θ pequeno, a curva estimada será um segmento de linha reta, passando, aproximadamente, *no meio da região*, cujo ponto central é o ponto estimado para $t = t_i$, $\mathbf{f}_\theta(t_i)$, e está posicionado no interior da curva geratriz, $\mathbf{f}(t_i)$. Desta forma, é

possível concluir que, em trechos com curvatura, o processo de estimação acarreta na obtenção de uma curva com raio r_θ *menor* que o da curva geratriz r [9].

Sob certas condições, pode ser mostrado [8] que o raio da curva estimada é igual a

$$r_\theta = r^* \ \frac{sen(\theta/2)}{\theta/2}. \tag{2.13}$$

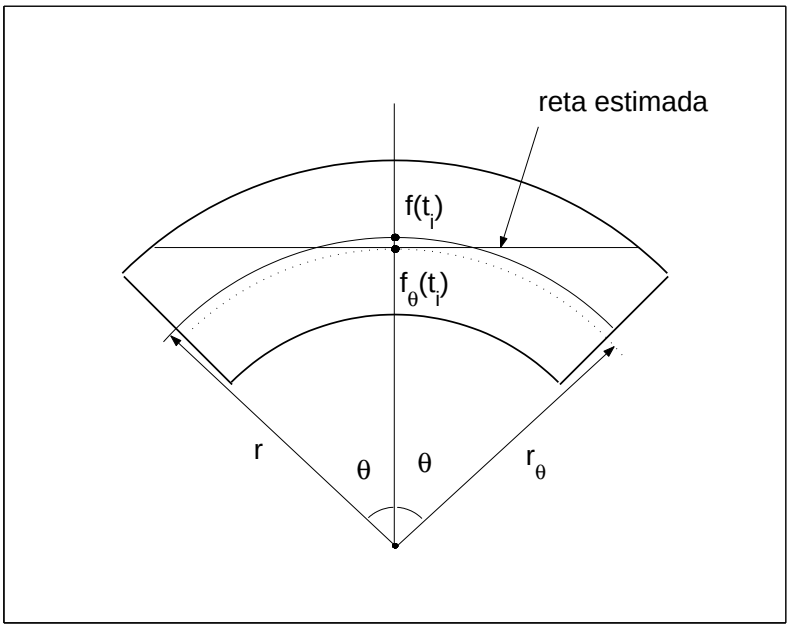


Figura 2.7: Modelo para estudo da tendência devida à estimação na curva principal definida por HS.

A tendência da curva principal foi objeto de estudo de outros trabalhos que serão abordados em seções subsequentes. A seção 2.1.4 apresenta o método de correção da tendência de estimação proposto por Banfield e Raftery [2].

Muito embora não tenham obtido uma prova formal da covergência do algoritmo ou da existência de uma curva principal para uma distribuição arbitrária dos dados, HS empregaram o algoritmo proposto na extração de curvas principais a partir de diversos conjuntos de dados reais e simulados, tendo alcançado a convergência todas as vezes.

A análise teórica do modelo de curva principal para um conjunto de dados é dificultada pela natureza não-paramétrica das curvas. Este assunto foi tratado

em diversos trabalhos posteriores ao de HS, os quais serão comentados nas próximas seções. No entanto, sob o ponto de vista prático, é mais simples operar com modelos não-paramétricos, especialmente quando os dados possuem dimensões elevadas.

Em [14] e [15] Duchamp e Stuetzle realizam um estudo teórico sobre curvas principais no plano cartesiano. No primeiro artigo, esses autores mostram que as curvas principais não se constituem num mínimo da função de distância, mas em pontos de sela desta função. No segundo trabalho, os autores mostram que as curvas principais podem ser obtidas, analiticamente, através da solução de um sistema de equações diferenciais. Para cada caso, há várias soluções, mostrando que, em geral, não há uma curva principal única. Além disso, eles provam que as curvas principais obtidas para uma distribuição, obrigatoriamente, se interceptam, o que seria uma propriedade análoga à ortogonalidade das componetes principais.

2.1.4 Correção da tendência de estimação

Os métodos não-paramétricos de ajuste de curvas empregados por HS, geralmente, produzem curvas que possuem um desvio com relação ao centro de curvatura.

A fim de corrigir a tendência de estimação, Banfield e Raftery [2], doravante denominados BR, propõem que, no Passo 1 do algoritmo de HS, a curva principal seja estimada conforme a seguinte equação

$$\mathbf{f}^{(j)}(t) = \mathbf{f}^{(j-1)}(t) + \delta^{(j-1)}(\mathbf{X}, t), \tag{2.14}$$

onde

$$\delta^{(j-1)}(\mathbf{X}, t) = E[\mathbf{X} - \mathbf{f}^{(j-1)}(t) | t_{\mathbf{f}^{(j-1)}}(\mathbf{X}) = t]. \tag{2.15}$$

De acordo com BR, o termo $\delta^{(j)}(\mathbf{X}, t)$ pode ser visto como uma medida da tendência da curva $\mathbf{f}^{(j)}$ em t. Seja $\mathbf{p}_i^{(j)} = \mathbf{x}_i - \mathbf{f}^{(j)}(t_i^{(j)})$ o resíduo da projeção do ponto $\mathbf{x}_i$ em $\mathbf{f}^{(j)}$. A tendência $\delta^{(j)}(\mathbf{X}, t)$ pode, então, ser definida como o valor esperado dos resíduos de projeção dos pontos que projetam na curva $\mathbf{f}^{(j)}$ em t. Os autores propõem que, no algoritmo para um conjunto de dados $\mathbf{X}$, os resíduos de projeção sejam usados em lugar dos dados a fim de obter-se $\mathbf{f}^{(j+1)}(t)$. Seja

$$\bar{\mathbf{p}}_i^{(j-1)} = \frac{\sum_{k=1}^{n} w_{ik}\mathbf{p}_k^{(j-1)}}{\sum_{k=1}^{n} w_{ik}} \tag{2.16}$$

a média ponderada dos resíduos de projeção dos pontos que projetam próximo de $\mathbf{f}^{(j-1)}(t_i^{(j-1)})$. Empregando $\bar{\mathbf{p}}_i^{(j-1)}$ a fim de estimar a tendência, um novo ponto da curva é obtido através da seguinte equação

$$\mathbf{f}^{(j)}(t_i^{(j-1)}) = \mathbf{f}^{(j-1)}(t_i^{(j-1)}) + \bar{\mathbf{p}}_i^{(j-1)}. \tag{2.17}$$

A Figura 2.8 mostra três curvas principais estimadas para um conjunto de dados em duas dimensões, empregando-se o algoritmo de HS. É importante notar que a tendência da curva é muito sensível com relação ao valor do coeficiente de suavidade, o qual, no algoritmo de HS deve ser definido pelo usuário. Nota-se também que, a medida que aumenta o valor do coeficiente de suavidade, a curva tende a tornar-se mais suave, no entanto, sua tendência aumenta.

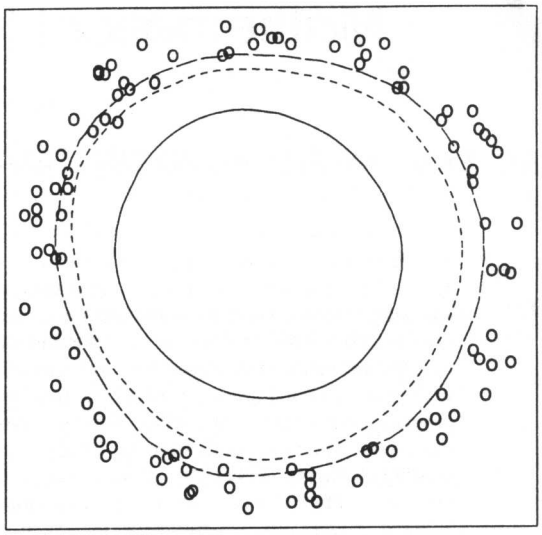


Figura 2.8: Curvas principais obtidas com o algoritmo de HS para eventos gerados a partir de uma distribuição uniforme de pontos distribuídos sobre uma circunferência com ruído aditivo Gaussiano. Cada curva foi obtida empregando-se um valor distinto de coeficiente de suavidade: 0,2 no caso da curva mais externa (tracejada); 0,3 a curva central (pontilhada) e 0,5 a interna (linha cheia). (extraída de [2]).

Na Figura 2.9 são mostradas as curvas principais obtidas com o emprego do algoritmo proposto por BR, para os mesmos dados, e empregando os mesmos valores

do coeficiente de suavidade usados na Figura 2.8. Note que, não só a tendência de cada curva diminuiu muito, como também o algoritmo se mostra muito menos sensível com relação à escolha do valor do coeficiente de suavidade.

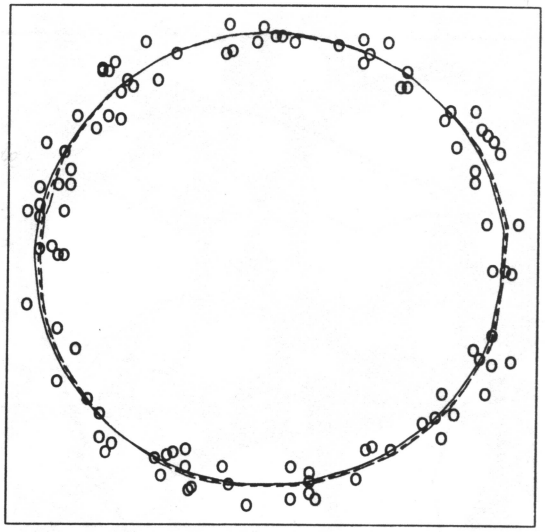


Figura 2.9: Curvas principais obtidas com o algoritmo de HS modificado por BR empregando os mesmos dados e valores do coeficiente de suavidade (0,2 curva tracejada, 0,3 curva pontilhada e 0,5 curva cheia) da Figura 2.8 (extraída de [2]).

BR também propõem uma modificação no método de regressão a fim de tornar a estimação da curva menos susceptível à presença de intrusos (*outliers*). Além disso, os autores estendem o algoritmo de HS visando à obtenção de curvas fechadas.

2.2 Definições alternativas

A definição de HS para curvas principais se baseia no princípio da auto-consistência. O modelo teórico de HS permite a obtenção de curvas suaves, não-parametrizadas e que não interceptam a si próprias. Conforme citado na seção anterior, o procedimento de HS para a extração de curvas para conjuntos de dados apresenta alguns problemas potenciais os quais serviram como motivação para trabalhos posteriores.

A seguir, apresentaremos importantes contribuições na área de curvas principais publicadas após o trabalho inicial de HS. No item 2.2.1, inicialmente, será

apresentada a definição de Tibshirani [16], a qual se baseia num modelo de mistura. Em seguida, serão apresentados alguns comentários sobre trabalhos apresentados por Chang e Ghosh [17] [18], os quais se baseiam na definição proposta por Tibshirani e estendem o conceito de curvas principais em mais de uma dimensão, definindo superfícies principais probabilísticas[9]. Em seguida, no item 2.2.2 apresentaremos o modelo de curvas principais de comprimento definido, proposto por Kégl et al. [20]. Estes autores provam que, para distribuições com momentos de segunda ordem finitos, as curvas principais definidas a partir do novo modelo garantidamente existem e são únicas. Na mesma linha do trabalho anterior, Sandilya e Kulkarni [21] obtêm resultados similares empregando, para distribuições com momentos de segunda ordem finitos, curvas principais com dobras limitadas. A seguir, no item 2.2.3, é apresentada a definição alternativa proposta por Delicado [22]. Também será abordado um dos seus trabalhos posteriores apresentado com Huerta [23]. Ainda neste item, faremos alguns comentários sobre o trabalho apresentado por Einbeck et al. [24], o qual se baseia na definição proposta por Delicado.

2.2.1 Modelos paramétricos

Vimos que, se os dados são gerados a partir do modelo aditivo (Equação 2.10), a curva $\mathbf{f}$, geralmente, não é uma curva principal de $\mathbf{X}$. A fim de resolver este problema, Tibshirani [16], propõe um novo modelo de definição dos dados. Assim como HS, ele, inicialmente, trata de curvas principais para distribuições e, em seguida, aborda o caso de $\mathbf{X}$ ser um conjunto de dados. Vejamos, então, o primeiro caso. Seja $\mathbf{X} = (\mathbf{X_1}, \ldots, \mathbf{X_d})$ uma variável aleatória com densidade $g_{\mathbf{X}}(\mathbf{x})$. A fim de definir uma curva principal, $\mathbf{f}(t)$, suponha que cada valor assumido pela variável aleatória $\mathbf{X}$ foi gerado em dois estágios sucessivos:

1. A variável latente t é gerada de acordo com uma distribuição $g_{\mathbf{T}}(t)$; e

2. A variável aleatória $\mathbf{X}$ é gerada a partir de uma distribuição condicional $g_{\mathbf{X}|T}(\mathbf{x}|t)$, cuja média, $\mu_{\mathbf{X}|T}$, é igual a $\mathbf{f}(t)$, sendo cada uma das componentes de $\mathbf{X}$, $(\mathbf{X_1}, \ldots, \mathbf{X_d})$, condicionalmente independentes dado t.

[9]Em [1], HS já haviam definido superfícies principais como uma extensão do conceito de curvas principais. Outro trabalho sobre o assunto foi apresentado por LeBlanc e Tibshirani [19].

Usando este modelo, a curva principal da distribuição $g_{\mathbf{X}}(\mathbf{x})$ pode ser definida como um contexto $\{g_{\mathbf{T}}(t), g_{\mathbf{X}|T}(\mathbf{x}|t), \mathbf{f}(t)\}$, satisfazendo as seguintes condições [16]:

1. $g_{\mathbf{T}}(t)$ e $g_{\mathbf{X}|T}(\mathbf{x}|t)$ são consistentes com relação a $g_{\mathbf{X}}(\mathbf{x})$, ou seja, $g_{\mathbf{X}}(\mathbf{x})$ é definido a partir do modelo de mistura entre $g_{\mathbf{T}}(t)$ e $g_{\mathbf{X}|T}(\mathbf{x}|t)$, $g_{\mathbf{X}}(\mathbf{x}) = \int g_{\mathbf{X}|T}(\mathbf{x}|t) g_{\mathbf{T}}(t) dt$.

2. $\mathbf{X_1}, \ldots, \mathbf{X_d}$ são condicionalmente independentes dado t.

3. $\mathbf{f}(t)$ é uma curva em $\Re^d$, parametrizada por $t \in I$, um intervalo fechado em $\Re$, satisfazendo $\mathbf{f}(t) = \mathbf{E}(X|T = t)$[10].

Conforme enfatizado pelo autor, esta nova definição envolve não apenas a curva $\mathbf{f}(t)$, mas a decomposição de $g_{\mathbf{X}}(\mathbf{x})$ em $g_{\mathbf{T}}(t)$ e $g_{\mathbf{X}|T}(\mathbf{x}|t)$.

Baseado nesta nova definição, Tibshirani propõe um método de extração de uma curva principal a partir de um conjunto de dados. Assuma que se dispõe de n observações da variável aleatória $\mathbf{X}_i = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, $i = 1, \ldots, n$, as quais estão relacionadas à variável latente não observada $t = t_1, \ldots, t_n$. Da mesma forma que definido para o caso de distribuições, assuma que t_i é gerado de acordo com a distribuição $g_T(t)$, que $\mathbf{x_i}$ é gerado conforme $g_{\mathbf{X}|t_i}$ e que $\mathbf{f}(t) = \mathbf{E}(\mathbf{X}|t)$. A forma de $g_{\mathbf{T}}(t)$ não é considerada, mas se assume que $g_{\mathbf{X}|T}(\mathbf{x}|t)$ pertence a uma família de distribuições cujos parâmetros $\Sigma(t)$ são desconhecidos. O autor propõe o emprego do algoritmo EM (expectation maximization) [3] na estimação de $\mathbf{f}(t)$ e $\Sigma(t)$. A função a ser maximizada é uma função de verossimilhança com uma componente de ajuste, garantindo a convergência do algoritmo.

Apesar da nova definição possuir vantagens teóricas com relação à definição de HS, na prática, o método proposto não produz resultados melhores que os obtidos através do método de HS [9]. Duas causas podem ser apontadas[18]:

1. Apesar da formulação proposta não apresentar a tendência relativa ao modelo, ela ainda é afetada pela tendência devida à estimação. Isto ocorre porque, assim como no método proposto por HS, a extração da curva

[10]Cada ponto da curva $\mathbf{f}(t)$ é a média condicional de $\mathbf{X}$, o que, em geral, não concorda com a definição da curva principal *auto-consistente* de HS. Veja detalhes em [16].

principal é realizada empregando-se um estimador não-paramétrico baseado em médias locais[11].

2. Como a nova definição não incorpora a propriedade de auto-consistência, não há como garantir que a curva principal definida por Tibshirani corresponde a um mínimo local da função de distância ou, em outros termos, não há como garantir que a curva otimiza o *erro médio quadrático (MSE) de reconstrução*.

Chang e Ghosh publicaram dois trabalhos [17] [18] abordando curvas e superfícies principais. Seus trabalhos se baseiam no modelo probabilístico de Tibshirani e em mapeamento generativo topográfico (GTM) [26]. As curvas resultantes são denominadas pelos autores curvas principais probabilísticas, as quais possuem duas vantagens importantes com relação ao modelo de Tibshirani: são, aproximadamente, auto-consistentes (essa propriedade é atingida no limite) e são facilmente extensíveis para a obtenção de superfícies principais, representando, segundo os autores, uma opção mais promissora que o método de HS e outros métodos baseados em mapas auto-organizados de Kohonen (SOM) [27] [28], GTM (generative topographic map) [26] e redes neurais auto-associativas [4] [29] [30].

2.2.2 Curvas principais ajustadas

Conforme vimos na seção 2.1.3, um dos problemas inerentes à definição de HS para curvas principais é que não é possível assegurar a existência de uma curva principal para uma distribuição arbitrária qualquer. Além disso, vimos que, de acordo com a definição de HS, uma curva principal é um ponto crítico da função de distância.

A fim de contornar estes problemas, Kégl, Krzyzak, Linder e Zeger [20] propõem uma nova definição de curvas principais possuindo um comprimento fixo e provam que curvas principais deste tipo sempre existem para distribuições com momentos de segunda ordem finitos. Especificamente, estes autores definem que uma curva de comprimento fixo L é uma curva principal para a distribuição h se ela é contínua e minimiza a função de distância $\Delta(h, \mathbf{f})$ dentre todas as curvas de comprimento menor ou igual a L.

[11] Tibshirani [16] propõe o emprego de splines (veja, por exemplo, [25]).

Além da nova definição, esses autores propõem um novo algoritmo para extração de curvas principais a partir de um conjunto de dados. Ao contrário dos algoritmos propostos por HS e Tibshirani, o algoritmo PLA (Poligonal Line Algorithm), proposto por Kégl, Krzyzak, Linder e Zeger, tem por objetivo obter uma curva poligonal[12] onde o número de vértices é muito menor que o número de eventos do conjunto de dados[13]. A Figura 2.10 mostra uma curva principal construída empregando-se o algoritmo PLA. A curva mais fina representa a curva geratriz; a curva mais espessa a curva principal; os eventos são representados por pontos que estão conectados por segmentos tênues à curva principal. Conforme pode ser observado, a curva principal coincide, quase exatamente, com a geratriz. A construção da curva principal é feita de forma incremental, ou seja, o número de segmentos de reta é aumentado até que a curva aproximada satisfaça um critério de convergência. Este novo algoritmo é bastante eficiente com relação aos algoritmos propostos por HS e Tibshirani. Além disso, ele possui diversos pontos em comum com o algoritmo de k-segmentos não suave, usado no desenvolvimento das aplicações propostas neste trabalho, o qual será apresentado no item 2.2.4.

Sandilya e Kulkarni [21] definem curvas principais com dobra total limitada e provam a existência deste tipo de curvas para uma distribuição com momentos de segunda ordem finitos. Considere uma curva poligonal $\mathbf{f}$, de vértices $\mathbf{v}_0, \ldots, \mathbf{v}_n$. Seja $a_i = \| \mathbf{v}_i - \mathbf{v}_{i-1} \|$ o comprimento do i-ésimo segmento e ϕ_i o ângulo entre os segmentos s_i e s_{i+1}. A dobra total $\mathbb{k}$ de uma curva poligonal $\mathbf{f}$ é definida como

$$\mathbb{k}(\mathbf{f}) = \sum_{i=1}^{n-1} \phi_i. \tag{2.18}$$

O algoritmo (teórico) proposto pelos autores se assemelha ao algoritmo descrito em [20], sendo que, naquele, a minimização da função de distância obedece à restrições relativas à dobra da curva.

[12]Uma curva aproximada por segmentos de reta, os quais são definidos por seus pontos extremos ou vértices.

[13]Empregando-se os algoritmos propostos por HS e Tibshirani, para cada evento do conjunto de dados, se obtém um ponto pertencente à curva principal que se deseja extrair.

Figura 2.10: Curva principal obtida para dados gerados a partir de uma distribuição arbitrária empregando o algoritmo PLA.

2.2.3 O modelo de Delicado

Sabe-se que a projeção de uma variável aleatória normal no hiperplano ortogonal à primeira componente principal possui a menor variância total dentre todas as projeções da variável em qualquer outro hiperplano. Delicado [22] propõe uma nova definição de curvas principais com base nesta propriedade da primeira componente principal. Seja $\mathbf{X}$ uma variável aleatória de distribuição h em $\Re^d$, tal que $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_d)$ e seja $\mathbf{x}$ um evento arbitário gerado conforme essa mesma distribuição. Considere $\mathbf{b}$ um vetor unitário em $\Re^d$ e $\mathbf{H}(\mathbf{x}, \mathbf{b})$ o hiperplano ortogonal a $\mathbf{b}$ passando por $\mathbf{x}$, ou seja

$$\mathbf{H}(\mathbf{x}, \mathbf{b}) = \{\mathbf{y} \in \Re^d \ : \ (\mathbf{y} - \mathbf{x})^T b = 0\}. \tag{2.19}$$

Considere, também, $\mu(\mathbf{x}, \mathbf{b})$ a média condicional de $\mathbf{X}$ no hiperplano $\mathbf{H}$, ou seja,

$$\mu(\mathbf{x}, \mathbf{b}) = \mathbf{E}[\mathbf{X}|\mathbf{X} \in \mathbf{H}(\mathbf{x}, \mathbf{b})]. \tag{2.20}$$

Delicado define $\mathbf{b}^* : \Re^d \rightarrow \Re^{d-1}$ como $\mathbf{b}^*(\mathbf{x}) = arg\,min\, \phi(\mathbf{x}, \mathbf{b})$, onde cada elemento de $\mathbf{b}^*$ é uma *direção principal* de $\mathbf{x}$ e $\phi(\mathbf{x}, \mathbf{b})$ é o ângulo entre $\mathbf{x}$ e $\mathbf{b}$. Este autor também define $\mu^* = \mu(\mathbf{x}, \mathbf{b}^*(\mathbf{x}))$. O conjunto $\Gamma(\mathbf{X})$ de *pontos principais*

orientados de X é o conjunto de pontos fixos de $\mu^* : \Gamma(\mathbf{X}) = \{\mathbf{x} \in \Re^d : \mathbf{x} \in \mu^*(\mathbf{x})\}$. Uma curva $\mathbf{f} : \Re^d \rightarrow I = [a, b]$ é uma *curva principal de pontos orientados* de $\mathbf{X}$ se, para todo $t \in [a, b]$, os pontos da curva $\mathbf{f}(t)$ são pontos principais orientados. Esta definição pode ser considerada como uma generalização da técnica de componentes principais visto que uma curva principal de pontos orientados maximiza a variância *total* dos dados projetados na curva, assim como a primeira componente principal é a direção que maximiza a variância das projeções dos dados.

Delicado prova que uma curva principal de pontos orientados sempre existe para distribuições com momentos de segunda ordem finitos e propõe um algoritmo para a extração de curvas principais. Em um segundo artigo publicado com Huerta[23], são apresentadas algumas considerações teóricas adicionais e um novo algoritmo.

Recentemente, Einbeck et al. [24] propuseram um novo algoritmo para extração de curvas principais cuja motivação se baseia no modelo proposto por Delicado. Os pontos fixos, μ^* e as direções principais, $\mathbf{b}^*$, ambos definidos por Delicado, são substituídos, respectivamente, por centros de massa locais e componentes principais locais. A curva principal é obtida conectando-se os centros de massa locais e, por isso, é denominada curva principal local. O método é bastante simples e intuitivo, no entanto, carece de uma análise teórica detalhada que demonstre que a curva obtida é, de fato, uma curva principal dos dados. Os autores apresentam diversas demonstrações em dados simulados e reais com excelentes resultados.

2.2.4 Algoritmo de k-segmentos não-suave

Neste trabalho, utilizamos o algoritmo de k-segmentos não-suave (*k-segments hard*) proposto por Verbeek, Vlassis e Kröse [7] na extração de curvas principais. O método se destaca, em relação aos demais, por apresentar uma maior robustez na estimação das curvas, menor susceptibilidade a mínimos locais e convergência prática garantida.

O método de k-segmentos não-suave propõe a construção da curva principal de forma incremental, ou seja, a curva principal é iniciada com apenas um segmento, sendo este número aumentado progressivamente. A cada inserção de um novo segmento, toda curva é novamente otimizada. A Figura 2.11 mostra a

seqüência de construção de uma curva principal para uma distribuição de eventos aleatória empregando o algoritmo de k-segmentos não-suave. Observe que a curva principal construída com o algoritmo de k-segmentos não-suave pode interceptar a si própria, contrariando uma das características básicas definidas por HS. Os resultados apresentados pelos autores [7] mostram que o algoritmo de k-segmentos não-suave consegue obter uma boa aproximação de curvas complexas, inclusive de uma espiral com duas voltas completas, enquanto que outros métodos, incluindo o algoritmo PLA, não demonstram a mesma habilidade.

Na inserção do primeiro segmento, todos os eventos do conjunto de dados são considerados, definindo-se um agrupamento cujo centro corresponde ao valor médio dos dados. Para este agrupamento, é estimado o primeiro segmento do método, o qual possui a direção da componente principal dos dados a ele pertencentes, e cujo comprimento vale 3/2 do desvio-padrão associado a esta componente.

Na inserção do segundo segmento, é realizado um teste, envolvendo todos os eventos do conjunto de dados, a fim de se definir qual é o melhor ponto a ser tomado como centro do agrupamento correspondente a este segmento. A fim de definir quais eventos pertencem ao novo agrupamento, é aplicado o algoritmo dos k-vizinhos mais próximos (*k-Means*) [3]. De posse deste agrupamento, o segundo segmento é determinado, seguindo o mesmo procedimento do primeiro.

Posteriormente, são calculadas as distâncias dos eventos aos segmentos mais próximos. Se, no processo de inserção deste segmento, algum evento antes associado ao primeiro segmento obtido muda de segmento, é repetido o processo de agrupamento, sendo os segmentos associados a este evento recalculados. Em seguida, realiza-se a interconexão dos segmentos, minimizando uma função de custo que considera os comprimentos dos segmentos e os ângulos entre cada segmento e seus segmentos adjacentes.

A inserção do terceiro segmento segue os passos anteriores e continua-se a inserção de segmentos até que um número máximo de segmentos tenha sido atingido ou o novo agrupamento possua menos que três pontos no seu interior. A Figura 2.12 ilustra o procedimento, onde k representa o número de segmentos já obtidos e k_max o número máximo de segmentos que se deseja obter.

O pseudo-código listado a seguir descreve os principais passos executados pelo

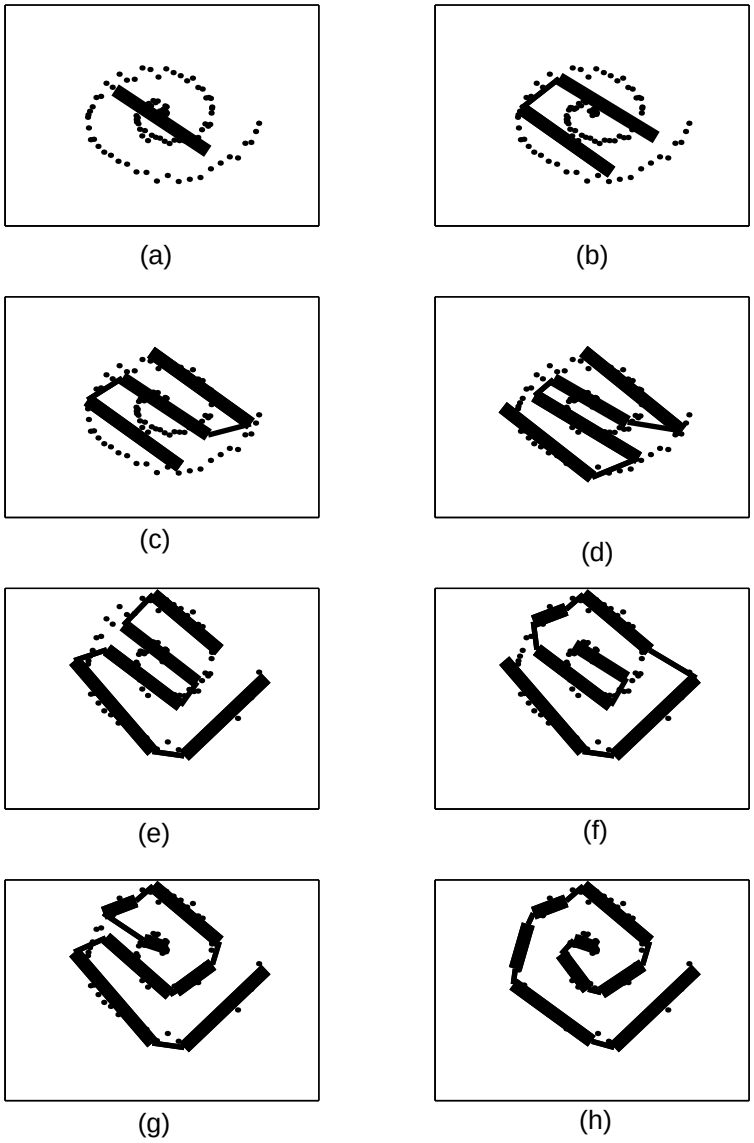


Figura 2.11: Seqüência de construção de uma curva principal, para um conjunto de pontos gerados a partir de uma distribuição arbitrária, empregando o método de k-segmentos não-suave. As linhas grossas representam os segmentos obtidos pelo algoritmo enquanto que as linhas finas representam junções que unem os segmentos. A primeira aproximação da curva é mostrada em (a) e trata-se de um segmento de reta cuja direção coincide com a da primeira componente principal de todo o conjunto de dados. A segunda aproximação, mostrada em (b) trata-se de uma curva poligonal formada por dois segmentos unidos por uma junção. O restante da seqüência de construção é mostrada nos gráficos seguintes, resultando numa curva poligonal composta por oito segmentos, conforme mostrado em (h).

algoritmo de k-segmentos não-suave durante a construção de uma curva principal. Considere k o número atual de segmentos que compõem a curva principal e k_max o número máximo de segmentos que se deseja obter.

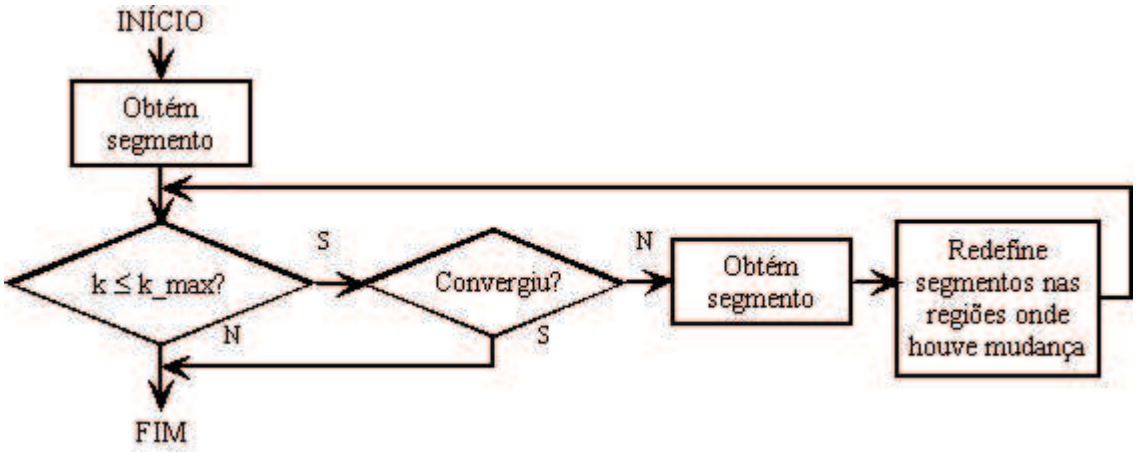


Figura 2.12: Fluxograma simplificado do algoritmo de k-segmentos não-suave.

```
10  k = 1;
20  Obtém o primeiro segmento;
30  Verifica a convergência;
40  Enquanto (k < k_max) e (não convergiu)
50     k = k + 1;
60     Obtem as regiões de Voronoi e armazena os eventos correspondentes na matriz vr;
70     Verifica qual é a maior região de Voronoi dentre as obtidas no passo anterior;
80     Determina o novo segmento considerando os eventos da região definida no passo
       anterior;
90     change = 1;
100    Enquanto change = 0
110       old_vr = vr;
120       Obtém, novamente, as regiões de Voronoi e armazena em vr;
130       Para i = 1 : k
140          Se, devido à inclusão do último segmento, houve mudança na i-ésima região,
             então
150             Recalcula os segmentos nas regiões onde houve mudança;
160          Fim (se)
170       Fim (para)
170       Se old_vr = vr,  então  change = 0;
180    Fim (enquanto change = 0)
190 Fim (enquanto (k < k_max) e (não convergiu))
```

As regiões de Voronoi citadas na linha 70 são compostas por eventos que estão mais próximos de um outro evento (centro da região) que de um dos segmentos que

compõe a curva principal. Ocorre uma mudança numa região de Voronoi (linhas 140 e 150) se pelo menos um evento que fazia parte dessa região passa a pertencer a uma outra região. Isto pode ocorrer, por exemplo, quando, após a inclusão de um segmento, um evento que anteriormente estava associado a uma região passa a não pertencer mais a ela.

O algoritmo de k-vizinhos mais próximos, empregado na construção da curva principal, possibilita o agrupamento dos dados em regiões de Voronoi, sendo que, a cada passo do algoritmo de k-segmentos não-suave, um novo segmento é inserido na maior das regiões obtidas. Aparentemente, este procedimento, o qual emprega agrupamentos locais, não parece produzir curvas que, de alguma forma, incorporam o conceito de auto-consistência definido por HS. De fato, enquanto que o algoritmo de HS busca obter pontos, aproximadamente, auto-consistentes (os quais unidos, gerarão uma curva, aproximadamente, auto-consistente), o algoritmo de k-segmentos não-suave busca inserir segmentos cujos pontos estejam mais próximos dos eventos da região que qualquer outro evento do conjunto de dados. No entanto, dois aspectos devem ser considerados. Os segmentos inseridos possuem a direção da primeira componente principal dos eventos contidos na região, sendo, portanto, auto-consistente com relação à regressão linear, conforme descrito na seção 2.1. Além disso, conforme Tarpey et al. [31], k pontos $\mathbf{y}_1, \ldots, \mathbf{y}_k$ são auto-consistentes se

$$\mathbf{y}_i = E[\mathbf{X}|\mathbf{X} \in V_i] \tag{2.21}$$

onde $V_1, \ldots, V_k$ são as regiões de Voronoi associadas aos pontos $\mathbf{y}_1, \ldots, \mathbf{y}_k$.

Além das características importantes como a robustez e a convergência garantida, o algoritmo requer a definição de poucos parâmetros (na prática, de apenas um, cujo valor *default* é bastante adequado). A chamada da rotina se dá por meio do comando

$$[vertices, edges, of, y] = k\_seg(X, k\_max, alpha, lambda),$$

onde as variáveis de entrada e saída e os parâmetros de entrada são definidos da conforme a seguir:

Variáveis e parâmetros de entrada:

X Matriz $n \times d$ de dados de projeto, onde n é a quantidade de dados e d é a dimensão dos dados.

k_max Número máximo de segmentos que deve compor a curva principal. O algoritmo deverá prosseguir inserindo segmentos até que o número máximo de segmentos seja atingido ou até que ocorra a convergência, o que ocorrer primeiro.

alpha O dobro do quadrado da distância esperada entre os eventos do conjunto de projeto e a curva. Muito embora este não seja um parâmetro opcional, ele não é empregado na extração da curva principal servindo, apenas, para a determinação de valores do vetor de saída *of*, o qual será descrito a seguir.

lambda Após a obtenção dos segmentos, o programa deve conectá-los, a partir de seus vértices, a fim de formar uma curva. Esta tarefa é realizada de forma a minimizar uma função de custo que considera as distâncias e os ângulos entre os vértices dos segmentos. O objetivo é ligar segmentos que estejam próximos e obter uma curva o mais suave possível. Se empregado o valor *default*, igual a 1, os dois objetivos – encurtar, ao máximo, a ligação entre segmentos e diminuir, tanto quanto possível, o ângulo formado pelas ligações entre segmentos – contribuirão, igualmente, para o valor da função de custo. Aumentando-se o valor de *lambda*, dá-se prioridade a curvas mais suaves, enquanto que, diminuindo-se seu valor, se prioriza a obtenção de curvas de menor comprimento.

Variáveis de saída:

vertices Contém as coordenadas dos vértices dos segmentos que compõem a curva principal extraída. Os pontos estão organizados em uma matriz $2k \times d$ onde k é o número de segmentos que compõem a curva e d é a dimensão dos dados.

edges Matriz $2k \times 2k$ que define como os vértices estão ligados a fim de formar a curva principal. De acordo com o tipo de conexão entre os vértices, os elementos da matriz *edges* assumem os seguintes valores : 0 - não há conexão entre os vértices; 1 - os vértices fazem parte de segmentos distintos mas estão

conectados; e 2 - os vértices são os pontos extremos de um segmento. Considere, por exemplo, uma curva principal formada por apenas dois segmentos, cuja matriz *vertices* = $[\mathbf{v_1}\ \mathbf{v_2}\ \mathbf{v_3}\ \mathbf{v_4}]$, de dimensão $d \times 4$, e matriz *edges* de dimensão 4×4 igual a

$$edges = \begin{pmatrix} 0 & 2 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 2 & 0 \end{pmatrix}.$$

Os índices dos elementos da matriz *edges* correspondem aos vértices listados na matriz *vertices*. Por exemplo, o elemento (2,3) indica que tipo de conexão existe entre o segundo ($\mathbf{v_2}$) e o terceiro vértices ($\mathbf{v_3}$) cujas coordenadas estão contidas na matriz *vertices*. Observando a matriz *edges* do exemplo é possível verificar que os vértices $\mathbf{v_1}$ e $\mathbf{v_2}$ formam um segmento, visto que os elementos (1,2) e (2,1) da matriz *vertices* contém o valor 2. Da mesma forma, os vértices $\mathbf{v_3}$ e $\mathbf{v_4}$ formam outro segmento. Como os elementos (2,3) e (3,2) contém o valor 1, conclui-se os vértices $\mathbf{v_2}$ e $\mathbf{v_3}$ unem os dois segmentos a fim de formar a curva, cujos extremos são os vértices $\mathbf{v_1}$ e $\mathbf{v_4}$.

of Matriz $k \times 3$ cujas linhas contêm informações referentes a cada curva intermediária obtida durante o processo de extração da curva final, a qual conterá um número máximo de segmentos. Em outras palavras, a primeira linha da matriz *of* contém informações relativas à primeira versão da curva principal procurada, a qual possui apenas um segmento; a segunda linha se refere à segunda curva obtida, a qual possui dois segmentos e assim sucessivamente, possuindo a última linha da matriz *of* apenas informações referentes à curva final extraída. A primeira coluna fornece a média das distâncias dos dados à curva (função de distância empírica, definida conforme a Equação 2.4), a segunda o valor do logaritmo da distância do ponto mais afastado da curva e a terceira o valor da função de custo.

y Os valores de cada linha da matriz $n \times (d+1)$ $\mathbf{y}$ estão relacionados aos n eventos da matriz de dados $\mathbf{X}$. A primeira coluna da i-ésima linha contém o valor do

índice de projeção[14] do i-ésimo evento da matriz $\mathbf{X}$. As demais colunas dessa linha contêm as coordenadas do ponto de projeção do i-ésimo evento da matriz X sobre a curva.

O algoritmo de k-segmentos não-suave foi implementado em MatLab, possui complexidade computacional $O(kn^2)$, onde n representa o número de eventos do conjunto de dados e encontra-se disponível para download na página eletrônica do primeiro autor, *http://staff.science.uva.nl/∼jverbeek/pc/index_en.html*.

Uma pequena modificação foi introduzida no algoritmo a fim de permitir a extração de curvas principais que, eficientemente, pudessem ser empregadas na tarefa de classificação. Além disso, foram introduzidas duas mudanças visando tornar o algoritmo mais robusto para a aplicação em questão e acelerar uma das rotinas. Essas modificações serão abordadas na Seção 3.3.

2.2.5 Análise comparativa dos métodos de extração de curvas principais

HS propõem um algoritmo de extração de curvas principais baseado no princípio da auto-consistência. Para cada evento pertencente ao conjunto de dados de treino um ponto pertencente à curva principal é extraído. Interligando-se esses pontos por meio de segmentos de reta encontra-se a curva principal procurada. Duas versões do algoritmo são propostas: uma emprega o método de regressão LOWRLS enquanto que a outra emprega splines. Ambas apresentam duas componentes de tendência: a primeira devida ao modelo aditivo empregado e a segunda devida ao método de estimação dos pontos pertencentes à curva principal procurada. Muito embora, teoricamente, seja possível cancelar os dois tipos de tendência, na prática isto não é simples, principalmente porque a tendência devida à estimação tende a ser muito mais expressiva.

[14] O índice de projeção é o comprimento, ao longo da curva, tomado desde o primeiro vértice (as extremidades da curva são definidas por dois vértices que só estão ligados a outro por um segmento; varrendo a matriz de edges, em ordem crescente de linha, o primeiro vértice da curva corresponde à primeira linha da matriz de edges, cujos elementos são iguais a 0 exceto por um deles, cujo valor é igual a 2) até o ponto de projeção.

Banfield e Raftery (BR) propõem uma modificação no algoritmo de HS que elimina a tendência devida à estimação e torna o algoritmo mais robusto com relação ao parâmetro de suavização empregado pela ferramenta de regressão.

Tibshirani propõe uma nova definição baseada num modelo de mistura. Apesar da definição de Tibshirani ser mais interessante com relação à de HS em alguns pontos (a nova definição se baseia num modelo paramétrico e, além disso, elimina a tendência relativa ao modelo), o algoritmo proposto não apresenta melhores resultados práticos que o de HS.

A definição proposta por Kégl et al. assegura a existência de curvas principais para distribuições com momentos de segunda ordem finitos. Além disso, o algoritmo proposto por esses autores é mais eficiente e produz melhores resultados que o proposto por HS, BR e Tibshirani.

A nova definição proposta por Delicado também assegura a existência de curvas principais com momentos de segunda ordem finitos. O algoritmo PCOP proposto por ele também é mais eficiente que o de HS e possui convergência garantida. Uma desvantagem, no entanto, é que as curvas extraídas podem possuir uma tendência.

O algoritmo de k-segmentos não-suave, empregado no desenvolvimento das aplicações propostas neste trabalho, apresenta várias características desejáveis como a convergência prática garantida e a robustez. Além disso, é um algoritmo eficiente, o qual, assim como o PLA (com o qual guarda diversas semelhanças) e o PCOP, extrai uma quantidade crescente de segmentos.

A Tabela 2.1 [18] resume as principais características dos algoritmos discutidos neste capítulo. As letras **S** e **N** foram usadas para indicar, respectivamente, se o algoritmo apresenta ou não a propriedade listada enquanto que n representa a quantidade de eventos do conjunto de dados empregado na extração das curvas.

2.3 Aplicações

Duas aplicações de curvas principais são relatadas por HS em [1]. A primeira se refere à correção das posições dos magnetos responsáveis pelo alinhamento dos feixes de partículas do colisionador linear SLC (Stanford Linear Collider). Esses magnetos foram distribuídos ao longo de uma curva suave, com aproximadamente 3

Tabela 2.1: Comparação dos principais algoritmos de obtenção de curvas principais. As siglas HS, BR, PLA, PCOP e k-seg se referem, respectivamente, aos algoritmos propostos por Hastie e Stuetzle, Banfield e Raftery, Kégl et al., Delicado e Verbeek et al.(k-segmentos não-suave).

Algoritmo	**HS**	**BR**	**PLA**	**PCOP**	**k-seg**
auto-consistência	S	S	N	N	N
tendência	S	S	N	S	N
quantidade de vértices	n	n	$< n$	$< n$	$< n$
complexidade computacional	$O(n^2)$	$O(n^2)$	$O(n^{\frac{5}{3}})$	$O(n^2)$	$O(n^2)$
existência garantida	N	N	S	S	S
modelo paramétrico	N	N	N	S	N

km de diâmetro e foram fixados com desvios em torno de $\pm$ 1 mm da posição ideal. Foi verificado que estes elementos não necessariamente deveriam estar posicionados exatamente sobre a curva ideal, mas ao longo de uma curva suave que permitisse a focalização dos feixes. Através do emprêgo da técnica de componentes principais foi possível obter a curva suave procurada e detectar quais magnetos, necessariamente, deveriam ser reposicionados.

Na segunda aplicação relatada por HS [1], uma empresa de extração de metais preciosos de placas eletrônicas descartadas deseja verificar qual, dentre dois laboratórios encarregados de analisar amostras dos lotes, detecta, em média, a menor quantidade de metal. A partir dos resultados encontrados por cada empresa na avaliação de 250 lotes distintos de placas, são construídas curvas principais, cuja análise permitiu identificar o laboratório procurado.

Banfield e Raftery [2] empregam curvas principais e morfologia matemática a fim de detectar e extrair os contornos de blocos de gelo em imagens de satélite. Outro trabalho que visa obter agrupamentos em imagens foi apresentado por Stanford e Raftery [32].

Kégl e Krzyzak [33] propõem uma extensão do algoritmo PLA [20] a fim de esqueletonizar caracteres em textos escritos a mão. De acordo com os autores, os resultados decorrentes do emprego da nova técnica são melhores que os obtidos com as técnicas tradicionalmente empregadas em processamento de imagens.

Chang e Ghosh [34] apresentam um classificador baseado em curvas princi-

pais. A partir de conjuntos de dados de projeto, é obtida uma curva principal para cada classe. A fim de classificar um evento pertencente a uma classe desconhecida, mede-se a distância euclidiana do evento às curvas principais representativas de cada classe, atribuindo-se o evento à classe mais próxima. Os autores descrevem o algoritmo utilizado na extração e apresentam diversos resultados obtidos com o emprego de bases de dados conhecidas. Em vários casos, os autores obtém eficiências de classificação melhores que as obtidas com métodos clássicos e, atribuem este sucesso à eficiência do método de curvas principais na extração de características para sinais cuja distribuição é não-esférica. As aplicações propostas neste trabalho são baseadas neste artigo.

2.4 Sumário

As curvas principais foram introduzidas por Hastie e Stuetzle (HS) em 1989 [1] como uma generalização não-linear da técnica de componentes principais. Definidas a partir do conceito de auto-consistência, elas são uma ferramenta capaz de mapear um espaço multidimensional $\Re^d$ num espaço unidimensional. Partindo do caso ideal, onde os dados são gerados a partir de uma distribuição conhecida, cada ponto da curva é definido como o valor esperado dos dados que nele projetam (conceito de auto-consistência). Além disso, conforme definido originalmente por HS, são curvas suaves e que não interceptam a si próprias.

Além de desenvolver uma teoria baseada numa distribuição de probabilidade, HS propõem um algoritmo capaz de extrair curvas principais a partir de um conjunto limitado de pontos. Como neste caso, não é possível aplicar, diretamente, a propriedade da auto-consitência a fim de definir cada um dos pontos da curva, os autores empregam técnicas de regressão linear local. Para cada ponto do conjunto de dados, um ponto da curva é calculado. Interligando-se convenientemente esses pontos, por meio de segmentos de linha reta, obtém-se uma curva poligonal cuja forma é *aprendida* a partir dos dados.

Duchamp e Stuetzle [15] [14] estudam curvas principais no plano e concluem que elas não são únicas e que, além disso, não são um mínimo mas pontos de sela da função de distância.

O algoritmo proposto por HS possui alguns problemas que são apontados pelos próprios autores, o que motivou outros pesquisadores a propor definições e algoritmos alternativos. Destacam-se os trabalhos de Banfield e Raftery [2], Tibshirani [16], Kégl et al. [20], Delicado [22] e Chang e Ghosh [17]. Banfield e Raftery propõem uma modificação no algoritmo de HS que elimina a tendência devida à estimação e torna o algoritmo mais robusto com relação ao parâmetro de suavização empregado pela ferramenta de regressão. Tibshirani propõe uma nova definição baseada num modelo de mistura. Apesar da definição de Tibshirani ser mais interessante com relação à de HS em alguns pontos (a nova definição se baseia num modelo paramétrico e, além disso, elimina a tendência relativa ao modelo), o algoritmo proposto não apresenta melhores resultados práticos que o de HS. A definição proposta por Kégl et al. assegura a existência de curvas principais para distribuições com momentos de segunda ordem finitos. Além disso, o algoritmo proposto por esses autores é mais eficiente e produz melhores resultados que os de HS. A nova definição proposta por Delicado também assegura a existência de curvas principais com momentos de segunda ordem finitos. O algoritmo proposto por ele também é mais eficiente que o de HS e possui convergência garantida. Uma desvantagem, no entanto, é que as curvas extraídas podem possuir uma tendência. Chang e Ghosh [17] apresentam uma nova definição e um novo algoritmo que permite a extensão de curvas principais em superfícies principais. Finalmente, Verbeek et al [7] [35] propõem dois algoritmos que, segundo os autores, apresentam melhores resultados que diversos outros métodos, incluindo o algoritmo PLA, proposto por Kégl et al.. Um desses algoritmos, o de k-segmentos não-suave, foi empregado no desenvolvimento das aplicações propostas neste trabalho. Ele apresenta várias características desejáveis como a convergência prática garantida e a robustez. Além disso, o algoritmo ainda disponibiliza diversas informações úteis ao acompanhamento da construção da curva e à verificação das distâncias entre os eventos de projeto e a curva.

Capítulo 3

Classificação de navios baseada em sinais de sonar passivo

Para um submarino de guerra, a identificação da classe de um navio de superfície ou até de outro submarino, a partir de sinais acústicos captados pelo seu sonar passivo, é de grande importância para o seu desempenho em ações de patrulha e ataque. Sistemas automáticos de classificação podem auxiliar o operador, aumentando a rapidez e a confiabilidade na tomada de decisões.

Neste capítulo é proposto um classificador de contatos para sinais de sonar passivo que utiliza a técnica de curvas principais. O classificador proposto é facilmente expansível para um número arbitrário de classes e apresenta um baixo custo computacional na fase de operação. Considerando como critério de decisão a menor distância euclidiana do espectro obtido a partir do sinal recebido à curva definida para cada classe, é obtida uma eficiência média igual a 96,7%. A apresentação dos resultados de classificação é procedida de uma análise da distribuição espacial dos dados, da verificação do impacto do número de segmentos que definem as curvas e do efeito da normalização da informação espectral na eficiência de classificação. Ademais, a robustez do classificador é analisada com base em diversas medidas de distância entre curvas.

3.1 Introdução

Usualmente, submarinos navegam de forma silenciosa, fazendo uso do SONAR (**SO**und **NA**vigation and **R**anging) passivo como seu principal equipamento de detecção. Além de dispor de diversos filtros que permitem ao operador selecionar as faixas de audio que ele deseja ouvir, os sistemas modernos de sonar realizam as análises DEMON (**DEMO**dulated **N**oise) e LOFAR (**LO**w **F**requency **A**nalysis and **R**ecording), através das quais o operador obtém informações adicionais sobre os ruídos recebidos. A análise DEMON é baseada na envoltória do sinal de cavitação emitido por meios dotados de propulsão e é empregado na estimação da velocidade de rotação do eixo propulsor e do número de pás [36]. A análise LOFAR corresponde a uma análise espectral do ruído na faixa de freqüência de operação do sistema [6]. Os espectrogramas produzidos – denominados *lofargramas* – apresentam raias espectrais que podem ser associadas à maquinaria de propulsão e à outras máquinas de grande porte em operação no interior dos navios que se deslocam nas proximidades.

Com o aumento da capacidade de processamento e armazenamento de dados e devido à importância estratégica do submarino em se manter oculto, os sistemas sonar se tornaram mais complexos, incluindo, além das ferramentas de visualização, módulos de detecção e classificação automáticos, os quais se baseiam nas análises dos ruídos observados e em informações previamente obtidas sobre os alvos de interesse. As indicações fornecidas pelo equipamento são consideradas pelo operador, o qual é o responsável pela decisão final sobre a detecção e a identificação dos alvos.

Os trabalhos publicados abordando a classificação de alvos baseada em sinais de sonar passivo são relativamente recentes. Rajagopal et al. [37] (1990) constroem um classificador especialista desenvolvido na linguagem Prolog. O classificador se baseia nos tons de cavitação gerados pela maquinaria de propulsão, em tons e seus harmônicos gerados pelo movimento de hélices, eixos, pistões e engrenagens e em tons gerados pelo deslocamento do navio. Os autores estudam as fontes típicas de geração de ruídos em navios, apresentando suas faixas de freqüência e o relacionamento entre os tons e as características discriminantes procuradas. A estrutura do classificador para quatro classes distintas é apresentada sem, no entanto, serem abordados os dados ou fornecidas as eficiências médias de classificação. Li et al. [38] (1995) apresentam o sistema EXPLORE, um classificador especialista baseado em

seis características extraídas da análise na freqüência dos ruídos emitidos por navios, tais como o número de raias espectrais e a freqüência do tom de maior potência. O critério de decisão é do tipo MAP (máximo a posteriori), sendo o valor da função de custo obtida pelo somatório ponderado dos valores das características. Segundo os autores, o desempenho do classificador é bastante afetado pela escolha dos pesos e valores dos parâmetros empregados no algoritmo. Os autores não descrevem o conjunto de dados ou detalham os resultados obtidos, no entanto, citam que, em uma série de simulações, o sistema pôde identificar e assimilar algumas novas classes apresentadas e apresentou uma eficiência média de classificação melhor que 75%.

Além das abordagens baseadas em classificadores especialistas, diversos outros autores propuseram classificadores baseados em informação espectral dos sinais acústicos detectados. A maioria dos trabalhos publicados emprega a densidade espectral dos sinais (amplitudes dos tons de freqüência obtidos através da transformada de Fourier dos sinais) em diversas abordagens, dentre elas as baseadas em classificadores estatísticos, filtros casados, redes neurais, redes neurais especialistas, técnicas de processamento de imagens e ferramentas estatísticas de representação, como curvas principais, no nosso caso. Outros trabalhos propõem a extração da informação espectral através do emprego de bancos de filtros wavelet, cujas saídas alimentam redes neurais.

Em 1991, Cottle e Hamilton [39] empregaram redes neurais a fim de realizar a detecção, a extração de características e a classificação de alvos. Seus resultados mostraram que as redes neurais podem ser empregadas de forma eficiente, principalmente, nas tarefas de detecção e classificação. No mesmo ano, Baran e Coughlin [40] (1991) propõem um classificador neural do tipo perceptrons em multicamadas (MLP) [41] com o objetivo de distingüir sinais emitidos por navios de superfície distantes de ruídos gerados por submarinos próximos. Os autores empregam informação espectral obtida a partir de sinais simulados, cujo modelo é descrito em detalhe.

Um classificador baseado em um modelo estatístico do tipo auto-regressivo (AR) é proposto Huang et al. [42] (1997). Inicialmente, o classificador extrai os pólos de um modelo AR de vigésima ordem da função de densidade espectral dos ruídos emitidos. Em seguida, é aplicada a transformada de Karhunen-Löeve a fim

de reduzir a dimensão dos dados. Os autores também empregam modelos AR de quarta ordem, dos quais são extraídos os pólos. Os autores não tecem comentários sobre o conjunto de dados ou abordam as técnicas de pré-processamento empregadas mas, apenas citam que obtiveram uma eficiência média de classificação igual a 96%, para quatro classes distintas de navios, quando os espectros são obtidos através da transformada rápida de Fourier (FFT) de 256 pontos.

Dois artigos recentemente publicados empregam filtros casados na classificação de sinais acústicos irradiados por navios. Souza-Filho e Seixas [43] (2001) empregam informação espectral obtida a partir dos ruídos irradiados por diversos navios pertencentes a quatro classes a fim de construir um classificador baseado em filtro casado, obtendo uma eficiência média de classificação igual a 96,5%. Silva e Seixas [44] (2004) empregam um conjunto de dados mais extenso que o anterior, o qual é composto pela informação espectral de 557 tons na faixa de DC a 3kHz, obtida a partir de ruídos gerados por navios pertencentes a oito classes distintas. Empregando a abordagem estocástica [45], os autores desenvolvem diversos classificadores para um número variável de componentes, obtendo uma eficiência média igual a 90,5%, quando se empregam as 450 componentes principais de maior energia.

Ribeiro-Fonseca e Correia [46] (1994) empregam sinais acústicos, cujo pré-processamento consiste, basicamente, na amostragem e janelamento do sinal empregando uma janela de Hanning, filtragem, cálculo da FFT, seguida pelo descarte da metade dos coeficientes, mantendo a informação relativa à faixa de DC a 2 kHz e, por fim, a redução do ruído através da média de 10 vetores consecutivos de coeficientes. São apresentados os resultados para três tipos distintos de classificadores, sendo o primeiro baseado no algoritmo dos k-vizinhos mais próximos [3], o segundo um classificador especialista baseado numa árvore de decisão e, o terceiro neural. Os autores verificam que o classificador baseado em rede neural apresenta resultados bastante superiores que os obtidos com os demais, mostrando-se, inclusive, muito mais tolerante com relação aos efeitos da adição de ruído aos sinais. De forma similar ao trabalho apresentado por Ribeiro-Fonseca e Correia, Rajagopal et al. [47] (1994) desenvolvem três tipos de classificadores, sendo um estatístico, o outro especialista e o terceiro neural, também concluindo que a eficiência do classificador neural é significativamente superior a dos demais. Ao contrário do trabalho ante-

rior, no entanto, este propõe que os classicadores sejam integrados, aumentando a eficiência do classificador final.

Neves Jr.[48] (1995) emprega ruídos emitidos por 16 navios pertencentes a quatro diferentes classes, além de amostras de ruído de fundo e ruído biológico, a fim de projetar uma classificador baseado numa rede neural MLP com treinamento usando *backpropagation*, a qual emprega como entradas características obtidas a partir de espectrogramas na faixa de freqüência de 0 a 400 Hz. O autor obtém um índice de acerto igual a 77% para o conjunto de treino e 60% para o conjunto de teste.

Em uma série de artigos mais recentes, Soares-Filho, Seixas e Calôba propõem diversos classificadores neurais. A informação espectral empregada consiste em 400 tons de freqüência, na faixa de DC a 2,87 kHz, para quatro classes distintas. Em [49] (2000) os autores avaliam o efeito do emprego da média de uma quantidade crescente de espectros na eficiência média de classificação e na capacidade de generalização do classificador. Quando espectros individuais são substituídos por médias de espectros pertencentes ao conjunto de treino, a eficiência média de classificação flutua, mas apresenta uma leve queda com o aumento da quantidade de espectros empregados no cálculo da média. Provavelmente, isto ocorre porque a estatística se torna mais pobre, prejudicando a capacidade de generalização do classificador. Os resultados para o conjunto de teste são distintos. Conforme esperado, verifica-se que a eficiência média de classificação cresce com o aumento do número de espectros empregados na média. Em [50] (2001) é comparado o desempenho de classificadores neurais cujas entradas sofrem redução na dimensão através de três métodos diferentes: análise de componentes principais (PCA), componentes principais não-lineares (NLPCA) e componentes principais de discriminação (PCD). Empregando apenas 3 componentes principais de discriminação, é obtida uma eficiência média de classificação igual a 93%. A fim de obter um valor próximo de eficiência (igual a 92%), os autores verificam que seriam necessárias 33 componentes principais ou 9 componentes principais não-lineares, concluindo, assim, que o classificador baseado em PCD é mais eficiente e mais compacto que os demais. Em [51] (2002) é discutido um método de detecção e aprendizado de uma nova classe durante a operação de um classificador neural. Empregando o mesmo conjunto de dados para quatro classes

de navios, os autores usam uma rede neural ART, treinada para identificar três das quatro classes, obtendo uma eficiência média de classificação igual a 87%. Quando os dados da quarta classe são apresentados, 45% deles são detectados pela rede como *outliers*. Se empregados para treinar uma nova rede ART para a quarta classe, a qual é incorporada à primeira rede, 63% dos dados da quarta classe são corretamente classificados, com uma redução mínima na eficiência de classificação das três primeiras classes (de 87% para 85%).

Abordagens baseadas em wavelets e redes neurais foram propostas por Ghosh et al. [52], (1991) Chen et al. [53] (1998) e Seixas et al. [54] (2001). Os três trabalhos propõem o emprego de um banco de filtros do tipo *wavelet* na fase de pré-processamento dos sinais acústicos, cabendo a uma rede neural a classificação. O objetivo do trabalho apresentado por Ghosh et al. é a identificação automática e a classificação de sinais oceânicos de curta duração obtidos por meio de um sonar passivo. Os autores empregam dados reais, sendo parte deles pertencentes à base governamental DARPA *standard data set I*. O vetor de características que servirá como entrada para uma rede neural é composto por 24 coeficientes *wavelet* além de um parâmetro que indica o tempo de duração do sinal. São implementados sete esquemas de classificadores empregando diferentes tipos de redes neurais[1]: Adaptive Kernal Classifier (ACK) [56], Rapid Kernel Classifier (RCK) [56], MLP, Radial Basis Functions (RBF) [57] e Learning Vector Quantization (LVQ) [58], além de um classificador baseado no algoritmo de k-vizinhos mais próximos (*k-nearest neighbor*) [3] e outro em redes Pi-Sigma [59]. O melhor resultado, 100% de eficiência, é obtido para redes ACK. No segundo trabalho são empregados dados obtidos a partir dos ruídos irradiados por três navios pesqueiros. São desenvolvidos três tipos de classificadores, dependendo do tipo de rede neural empregada – MLP, AKC ou AKC modificada (rede AKC com raio fixo) – obtendo-se eficiências médias de classificação de até 96%. Seixas et al. empregam dados reais de quatro classes distintas de navios. A classificação é realizada por uma rede neural MLP. São discutidos dois métodos de normalização, obtendo-se uma eficiência média melhor que 94%.

[1]Em [55] são revistos diversos tipos de redes neurais aplicáveis ao problema de classificação de sinais com características espectrais variantes no tempo, sendo usados sinais obtidos por sistemas de sonar passivo a fim de ilustrar os conceitos apresentados.

Alguns trabalhos tratam do emprego de estatística de ordem superior na caracterização e classificação de sinais de sonar passivo. Regazzoni et al. [60] (1994) analisam o emprego de estatística de terceira ordem, concluindo que as informações obtidas, tanto em termos de freqüência como em fase podem caracterizar os ruídos detectados, embora eles não tenham desenvolvido um classificador específico para tal fim. Em [61] (1995), é estudada a viabilidade de obtenção de informações relevantes aplicadas à tarefa de classificação de sinais de sonar passivo empregando estatística de ordem superior. São analisados espectros de ordem superior (HOS - Higher Order Spectra) e apenas espectros de fase (POS - Phase Only Spectra), o qual é obtido descartando-se a informação de amplitude contida na análise HOS, tendo os autores concluido que, através da análise espectral de ordem superior HOS, não se obtém informação relevante não disponível na densidade espectral (segunda ordem) dos sinais e que pouca informação relevante adicional pode ser obtida através dos espectros de fase, POS. Pflug et al. [62] analisam segmentos de gravações realizadas durante intervalos contínuos de 30 minutos, em local próximo ao porto de San Diego, empregando um array vertical de quatro hidrofones instalados a profundidades que variam entre 116 e 192 metros. O ruído foi gravado a uma taxa de 1500 amostras/segundo e os sinais foram extraídos empregando-se uma janela de 1 segundo de duração. As análises visam verificar a estacionaridade e a gaussianidade dos ruídos gravados em ambiente costeiro, através da análise dos valores médios da variância, do coeficiente de assimetria (*skewness*) e da curtose. O estudo conclui que os ruídos detectados são, em geral, gaussianos, exceto por alguns períodos, durante os quais ocorrem variações nos valores da curtose devido à passagem de navios a uma distância menor ou igual a 1 km do array. O teste de Kolmogorov-Smirnov (K-S) aplicado a dois dos canais revela diversos períodos de não-estacionariedade. O estudo também mostra que os sinais detectados por canais adjacentes são correlacionados, com o coeficiente de correlação aumentando, em módulo, nos períodos onde se observam os maiores desvios da gaussianidade.

Sinais de sonar passivo também têm sido empregados na monitoração e na classificação de espécies marinhas e fenômenos geofísicos tais como abalos sísmicos. Ghosh et al. [63] [64] [65] empregam redes neurais e uma combinação de coeficientes wavelet e espectrais de um conjunto de dados de sons de animais marinhos e de gelei-

ras. Howell, Wood e Koksal [66] propõem um classificador baseado em redes neurais híbridas a fim de implementar um classificador compacto, o qual faria parte de um sistema embarcado como uma bóia ou um sistema autônomo. Os autores empregam dados reais de várias espécies além de sons emitidos por navios, armas submarinas e fenômenos naturais, divididos em 8 classes. Na fase de pré-processamento são obtidas as informações espectrais dos sinais as quais alimentam uma rede do tipo SOM (self-organizing map) seguida de uma rede MLP. Segundo os autores, este tipo de configuração permitiu obter bons resultados de classificação com uma rede bastante mais compacta do que uma única rede MLP.

Ghosh e Tumer[67] propõem métodos de integração de diversos classificadores neurais a fim de aumentar a eficiência de classificação e a robustez do classificador. Três métodos são comparados: adoção da classe mais votada, aplicação de um procedimento baseado em médias ou estimação de densidades. Segundo os autores, a integração melhora a eficiência de classificação além de prover meios para a detecção de intrusos (*outliers*) e a redução de falso alarme.

Apesar da eficiência já comprovada das redes neurais para lidar com problemas complexos como a classificação de sinais de sonar passivo, a absorção de novas classes em tempo de operação (ou em um tempo curto) ainda é um problema. Em um trabalho recentemente publicado, Costa et al. [68] (2004) analisam a aplicação de redes especialistas no projeto de um sistema automático de classificação de sinais de sonar passivo. Empregando informação espectral obtida a partir de dados reais pertencentes a três classes distintas de navios, o sistema, inicialmente composto por três redes neurais especialistas, apresenta uma eficiência média de classificação igual a 93%. É simulada a inclusão de duas novas classes, uma ou outra, sem que ocorra o retreinamento da rede. Neste caso, obtém-se uma pequena redução na eficiência, que passa para 89%. Após o retreinamento de toda a rede, verifica-se um aumento na eficiência média para 91%.

Outra abordagem interessante é apresentada por Thyagarajan e Nguyen [69] (1997), os quais propõem um classificador baseado na análise da textura de lofargramas obtidos para dados pertencentes a oito classes de sinais. O pré-processamento dos sinais envolve a aplicação de um filtro QMF de Johnston [70] e o emprego da transformada wavelet Daubechies' biortogonal [71]. É obtida uma eficiência média

de classificação igual a 99,99%.

O emprego de curvas principais ao problema de classificação de sinais de sonar passivo foi tratado por Alvear et al. [72] (2001). A abordagem empregada pelos autores, no entanto, é distinta da que será discutida nesta dissertação. Nela os autores empregam apenas uma curva principal que irá representar a informação espectral das quatro classes empregadas. Os resultados apresentados, ainda preliminares, dão conta de uma eficiência de classificação igual a 37,6%.

Neste capítulo, é proposto um classificador de contatos baseado em curvas principais para oito classes distintas de navios. Inicialmente, na seção 3.2, é apresentado o conjunto de dados experimentais utilizado na implementação e teste do classificador. Posteriormente, na seção 3.3, são apresentados os detalhes de implementação do classificador. A seguir, na seção 3.4, são mostrados os resultados obtidos com o emprego do classificador baseado em curvas principais, os quais serão comparados com os resultados alcançados por outros autores, empregando abordagens distintas, tais como filtros casados ou redes neurais. Na seção 3.5 são comparados resultados obtidos empregando-se o mesmo conjunto de dados e técnicas distintas de classificação. Finalmente, na seção 3.6 são sumarizados os assuntos tratados no capítulo.

3.2 O conjunto de dados

O conjunto de dados utilizado neste trabalho consiste nos ruídos irradiados por 25 navios, pertencentes a 8 classes distintas, durante 263 corridas de prova realizadas na raia acústica da Marinha do Brasil, situada em Arraial do Cabo, Rio de Janeiro. A cada corrida, um navio percorre a raia, deslocando-se em condições de maquinaria e operação constantes, sendo que cada navio realiza diversas corridas, cada uma operando em uma condição operativa diferente. Os ruídos emitidos pelos navios, juntamente com o ruído ambiente e as suas múltiplas reflexões no assoalho submarino, são detectados por um hidrofone posicionado a uma profundidade em torno de 45 metros, conforme esquematizado na Figura 3.1.

Esses sinais foram digitalizados com uma freqüência de amostragem de 22,05 kHz, utilizando 16 bits de resolução para as amplitudes. O sinal adquirido foi pré-

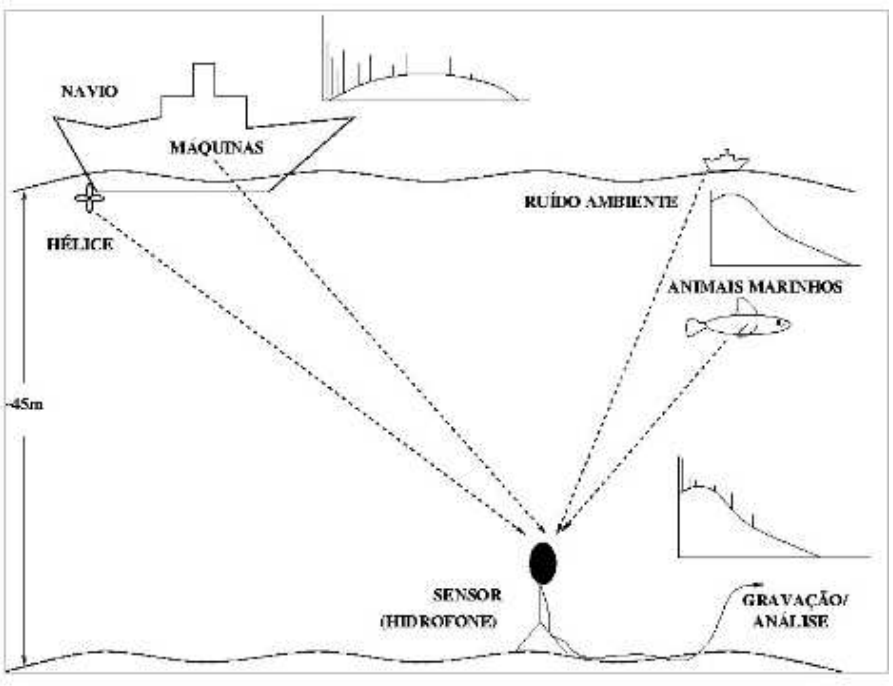


Figura 3.1: Obtenção dos ruídos irradiados por um navio durante uma corrida em uma raia acústica.

processado, visando obter seu espectro e realçar as características relevantes à tarefa de classificação, em especial, os picos espectrais, os quais estão relacionados a tons e harmônicos produzidos pelas máquinas em operação no interior das embarcações. A escolha da cadeia de pré-processamento, mostrada na Figura 3.2, foi realizada conforme estudo detalhado apresentado em [6].

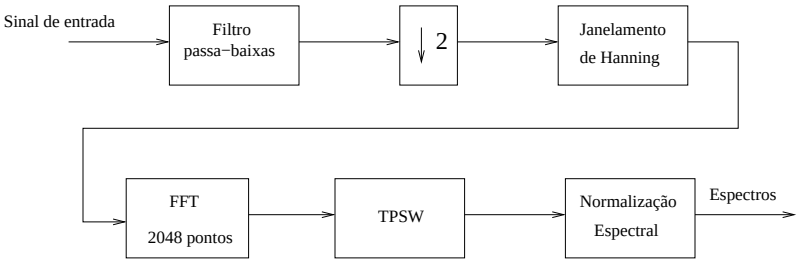


Figura 3.2: Cadeia de pré-processamento para obtenção dos espectros.

Inicialmente, cada corrida foi dividida em janelas de 4096 amostras sem superposição, o que corresponde a trechos com duração de aproximadamente 186 ms. Cada janela foi submetida a uma filtragem passa-baixa por um filtro de Chebyshev de oitava ordem do tipo I, seguida por decimação de um fator de 2, reduzindo a

taxa de amostragem para 11,025 kHz. As amostras resultantes sofreram um janelamento de Hanning e o módulo de sua transformada rápida de Fourier foi calculado. Dos 2048 pontos resultantes, apenas 557 foram considerados, de forma a cobrir a faixa de freqüência de 0 a 3 kHz, a qual contém informações relativas ao navio e às máquinas em operação no seu interior. Em seguida, os pontos espectrais foram submetidos ao algoritmo TPSW (*Two-Pass Split-Window*) [73], o qual provê uma estimativa da média local, a qual está relacionada ao ruído de fundo do ambiente de medição. Este algoritmo consiste no cálculo, em duas etapas, de uma média local que utiliza uma janela de largura n com uma fenda central de largura p, conforme mostrado na Figura 3.3. Uma primeira convolução desta janela com o espectro estima a média local, a qual corresponde à estimativa da média dos pontos vizinhos a cada ponto. A seguir, as amostras do espectro que excedem um limiar, dado pela média local multiplicada por um fator fixo a, são substituídos por suas médias locais, e uma nova convolução envolvendo o espectro resultante e a janela inicial é realizada. Cada amostra do espectro original é então dividida pelo ruído de fundo estimado [74]. A Figura 3.4 mostra a amplitude do sinal extraído de uma corrida arbitrária, antes e depois do processamento pelo algoritmo de TPSW. Finalmente, as amostras são normalizados pela sua média local, fornecendo os dados de entrada para o classificador. A Figura 3.5 mostra o espectrograma obtido a partir dos sinais mostrados na Figura 3.4. A escala horizontal representa a freqüência; a vertical, o tempo; enquanto as raias espectrais de maior intensidade correspondem aos níveis de cinza mais pronunciados. A Tabela 3.1 contém a quantidade de espectros que compõem os conjuntos de dados de cada classe.

Tabela 3.1: Quantidade de espectros que compõem os conjuntos de dados de cada classe.

Classe A	**Classe B**	**Classe C**	**Classe D**	**Classe E**	**Classe F**	**Classe G**	**Classe H**
2430	3432	4795	3072	7070	2934	2142	3392

Os espectros disponíveis foram divididos em dois conjuntos disjuntos: o conjunto de projeto e o de teste. O primeiro foi empregado na extração das curvas principais e nas diversas análises apresentadas nos itens 3.2.2, 3.2.3 e 3.3.5, enquanto que o segundo foi utilizado na avaliação da capacidade de generalização do classificador. Dado que o algoritmo de extração das curvas possui um razoável custo

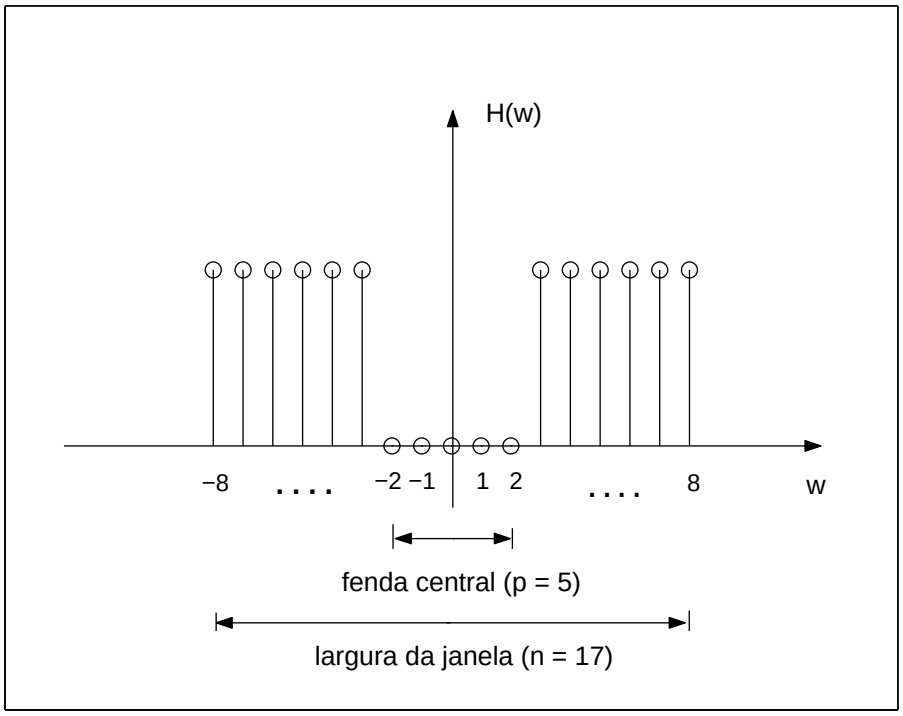


Figura 3.3: Janela empregada no algoritmo TPSW.

computacional, o número de espectros utilizados na extração da curva de cada classe foi restrito a um valor em torno de 1000. Como o número de espectros disponíveis para cada classe é de, no mínimo, 2174, e, no máximo, 7076, a divisão foi realizada de forma que para cada espectro do conjunto de projeto, K espectros consecutivos foram destinados ao conjunto de teste, onde K assumiu valores inteiros entre 2 e 6, de acordo a classe em questão. No caso do emprego de médias de dois espectros, os conjuntos são reduzidos à metade dos originais. A Tabela 3.2 mostra a divisão de espectros empregada no desenvolvimento do classificador. O emprego de médias de espectros leva a melhores resultados de classificação, conforme discutido em [49], visto que existe uma tendência à redução do ruído de fundo presente nos espectros. No entanto, como se pretende manter a estacionariedade dos dados [6] e se deseja obter um classificador rápido, apenas a média de dois espectros foi considerada, recaindo numa janela de observação de largura igual a 372 ms.

Antes de abordar a implementação do classificador, procedemos algumas análises estatísticas, visando verificar os tons de maior amplitude dos espectros de cada classe e caracterizar a distribuição espacial dos espectros das classes, a fim de avaliar a complexidade do processo de classificação. Os resultados da análise espa-

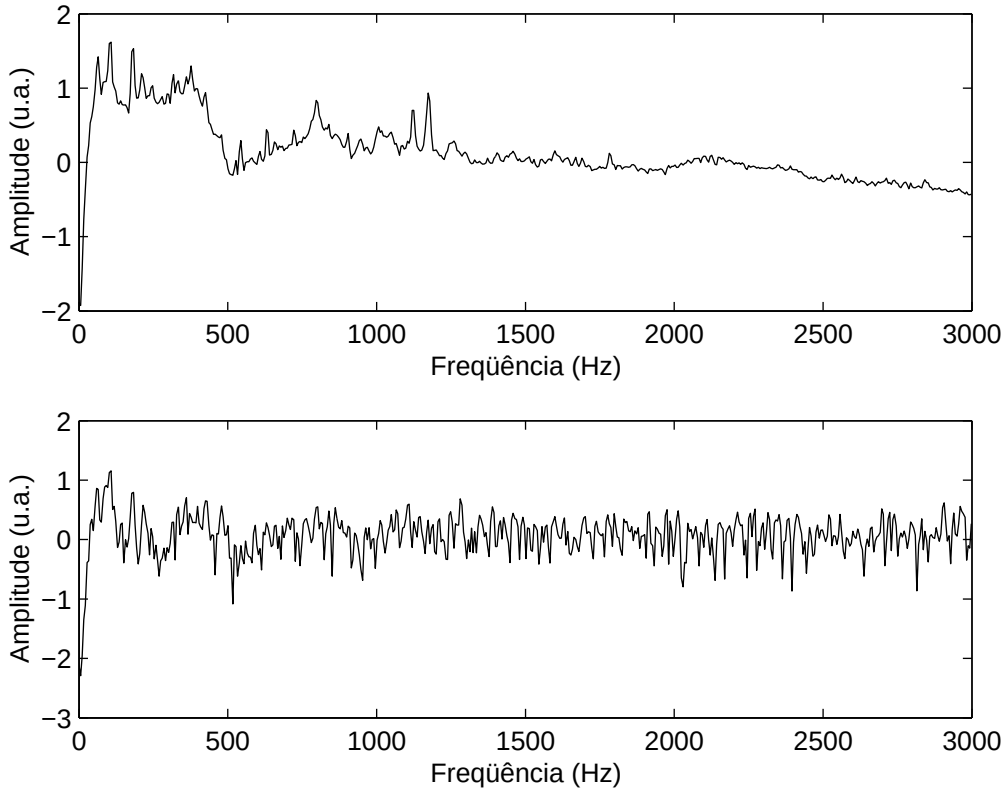


Figura 3.4: Amplitude do sinal antes do pré-processamento pelo algoritmo de TPSW (acima) e após o emprego do algoritmo de TPSW (abaixo).

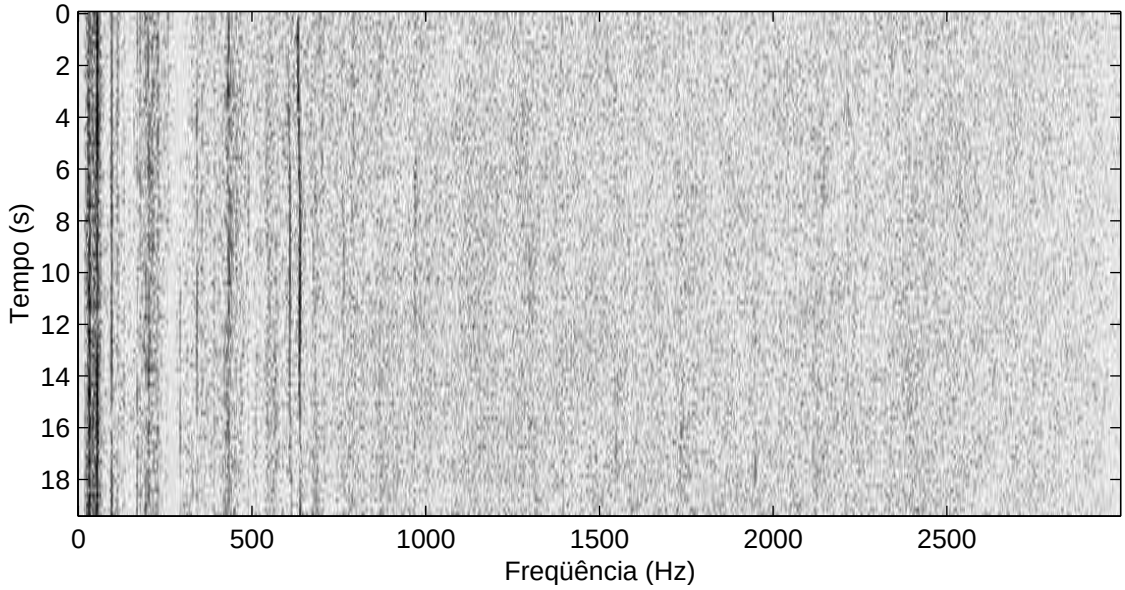


Figura 3.5: Espectrograma típico obtido a partir dos sinais adquiridos experimentalmente.

cial da distribuição dos espectros serão baseadas em dois indicadores: as distâncias entre os centros das classe e as médias das distâncias entre os espectros das diversas

Tabela 3.2: Quantidade de espectros que compõem os conjuntos de projeto e de teste de cada classe.

	Classe A	***Classe B***	***Classe C***	***Classe D***	***Classe E***	***Classe F***	***Classe G***	***Classe H***
Conj.Proj.	810	858	959	1024	1010	978	1071	848
Conj.Teste	1620	2574	3836	2048	6060	1956	1071	2544

classes.

3.2.1 Análise estatística dos dados

A Tabela 3.3 mostra os índices dos dez tons de freqüência de maior intensidade, presentes nos espectrogramas de cada classe[2], dispostos em ordem crescente de intensidade. A fim de obter esses tons, inicialmente, foram obtidas as médias dos dados, considerando todos os espectros de uma classe. Em seguida, foram selecionados os tons que apresentaram os maiores valores médios. Os dados contidos na Tabela 3.3 foram dispostos, graficamente, conforme mostrado na Figura 3.6, a fim de facilitar a visualização da distribuição dos tons, em função das classes. É possível observar que, embora esses tons se concentrem nas faixas mais baixas de freqüência (o tom de maior índice presente na Tabela 3.3 é igual a 159, o que corresponde a um valor de freqüência igual a 856 Hz), é possível supor que a presença de alguns tons ou a sequência dos tons mais intensos possam caracterizar uma determinada classe de navios, permitindo a discriminação das classes. Por exemplo, é possível verificar que, dentre os tons mais intensos irradiados pelos navios da classe A encontra-se o tom de índice 140, correspondente à freqüência de 754Hz. Esse tom não aparece como sendo um dos mais intensos de nenhuma outra classe. Muito embora esta análise seja simplificada, é possível supor que os navios da classe A possuam boa discriminação. As classes G e H, no entanto, apresentam tons intensos apenas para freqüências baixas, sendo parte desses tons também irradiados pelos navios de quase todas as outras classes. Analogamente, é possível supor que seja difícil discriminar os navios dessas classes do que os pertencentes as demais classes.

Adicionalmente, buscou-se estimar as funções de densidade de probabilidade

[2]Os índices dos tons de freqüência assumem valores na faixa de 1 a 557, correspondendo à faixa de freqüência de 0 a $\sim 3kHz$.

Tabela 3.3: Índices dos dez tons de freqüência de maior intensidade, na média, irradiados por navios das diversas classes.

	Classe A	**Classe B**	**Classe C**	**Classe D**	**Classe E**	**Classe F**	**Classe G**	**Classe H**
1	21	23	17	12	12	45	23	17
2	20	24	18	13	13	46	24	18
3	13	22	19	35	2	16	16	19
4	12	25	16	24	1	57	19	16
5	15	26	2	23	122	17	22	23
6	14	21	68	57	159	56	20	20
7	22	35	67	11	121	15	21	22
8	140	27	69	34	158	51	17	12
9	16	28	15	22	125	23	18	21
10	23	20	13	25	123	20	15	24

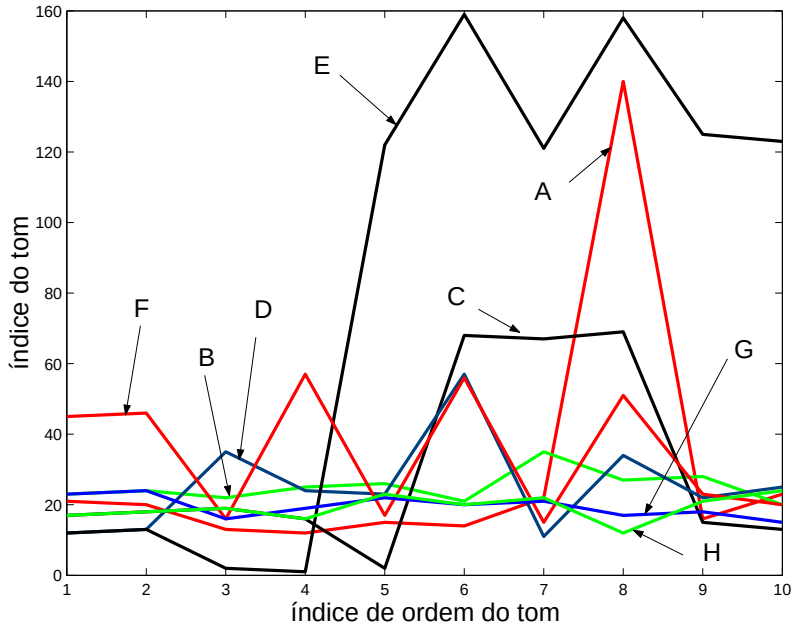


Figura 3.6: Índices dos dez tons de freqüência de maior intensidade, na média, irradiados por navios das diversas classes. O eixo dos x serve como índice de ordem de intensidade dos tons, assim, para $x = 1$ verifica-se, para todas as classe, os valores dos índices dos tons mais intenso, correspondente aos valores da primeira linha da Tabela 3.3. Analogamente, para $x = 2$, verificam-se os valores contidos na segunda linha da tabela citada.

(fdp) de diversos tons listados na Tabela 3.3. Neste trabalho, limitamo-nos a apresentar a forma típica das fdp desses tons e a discutir, em termos gerais, o ajuste (*fitting*) das fdp de tons irradiados por navios pertencentes a algumas classes, a fim de sustentar discussões que serão apresentadas na Seção 3.4. Resultados adicionais

poderão ser encontrados em [75].

A Figura 3.7 apresenta o ajuste da fdp do tom de índice 12, referente à classe A. O histograma mostra uma função não simétrica, de crescimento abrupto e queda mais suave, apresentando uma cauda à direita. Este padrão foi observado para vários tons de diversas das classes estudadas, tendo sido ajustado, com sucesso[3], empregando-se a função

$$y = \left(\frac{1}{1 + e^{(x - PAR(1))}}\right) * (PAR(2) + PAR(3) * x + PAR(4) * x^2 + PAR(5) * x^3), \quad (3.1)$$

a qual é formada pelo produto de uma função logística e um polinômio do 3° grau. A primeira função possibilita o crescimento abrupto verificado. PAR(1) a PAR(5) são os parâmetros da função.

A Figura 3.8 mostra a melhor tentativa de ajuste do tom de maior intensidade (na média), dentre os emitido por navios da classe E. A validação desse ajuste, através do teste do chi-quadrado, indica que a função empregada no ajuste, não adere aos dados, sendo possível rejeitar o modelo proposto. Diversas funções foram empregadas, sem que se tenha conseguido um ajuste validado pelo teste de aderência do chi-quadrado. Este fato também ocorreu durante as tentativas de ajuste de diversos outros tons da classe E, assim como para diversos tons das classes A e C. Vários tons da classe B, listados na Tabela 3.3, foram ajustados com sucesso, entretanto, foi necessário empregar uma função mais complexa, contendo sete parâmetros.

Analisando-se os espectros pertencentes às classes A, B, C e E, verifica-se que eles possuem padrões variados e alguns são muito ruidosos, não sendo, inclusive, possível perceber em alguns as raias espectrais que caracterizam o ruído emitido por um navio. A Figura 3.9 mostra alguns espectrogramas obtidos durante as corridas de navios pertencentes à classe E. Os dois primeiros apresentam padrões parecidos, enquanto que o terceiro apresenta um padrão distinto. O quarto deles é bastante ruidoso, não possuindo raias espectrais bem definidas, ao contrário dos demais. A diversidade de padrões dos espectros, indicando que as classes A, B, C e E possuem

[3]A validação do ajuste foi realizada empregando-se o teste do chi-quadrado (χ^2), para um grau de significância (α) igual a 0,05.

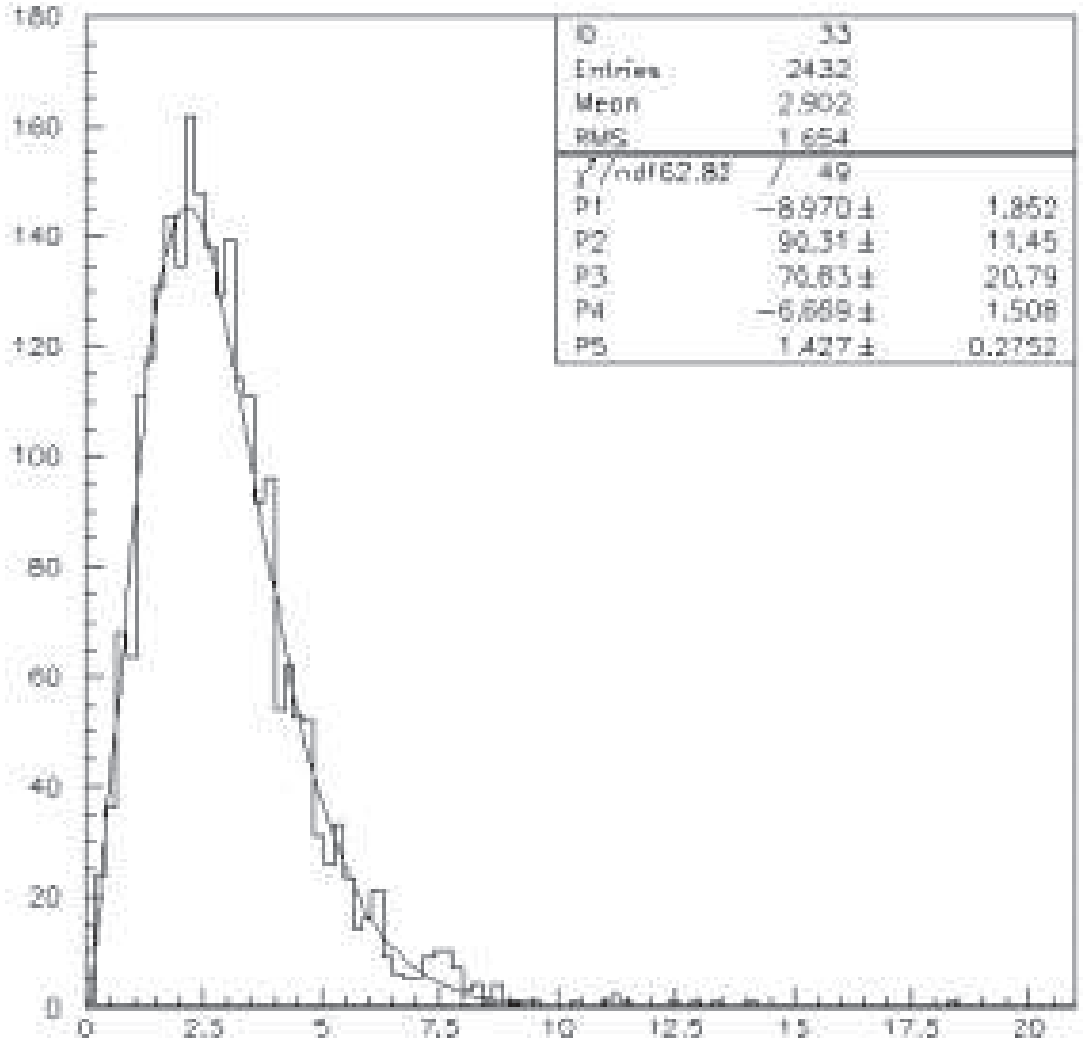


Figura 3.7: Histograma e ajuste da fdp para o tom de freqüência de índice 12 da classe A. O ajuste é validado pelo teste de aderência do chi-quadrado, uma vez que o valor do χ^2 obtido é igual a 62,8 (para 49 graus de liberdade), inferior ao valor limite do χ^2, o qual é igual a 66,3.

navios com características distintas, além da presença de espectros muito ruidosos, parecem explicar a dificuldade de ajuste das funções de densidade de probabilidade associadas aos tons pesquisados.

3.2.2 Distâncias entre os centros das classes

Através da avaliação da distância entre os centros das classes [4], é possível ter uma idéia de proximidade e, em conseqüência, produzir indicativos de quais clas-

[4]O centro de uma classe é obtido como a média do processo estocástico, considerando como suas realizações os espectros que compõem o conjunto de projeto da classe.

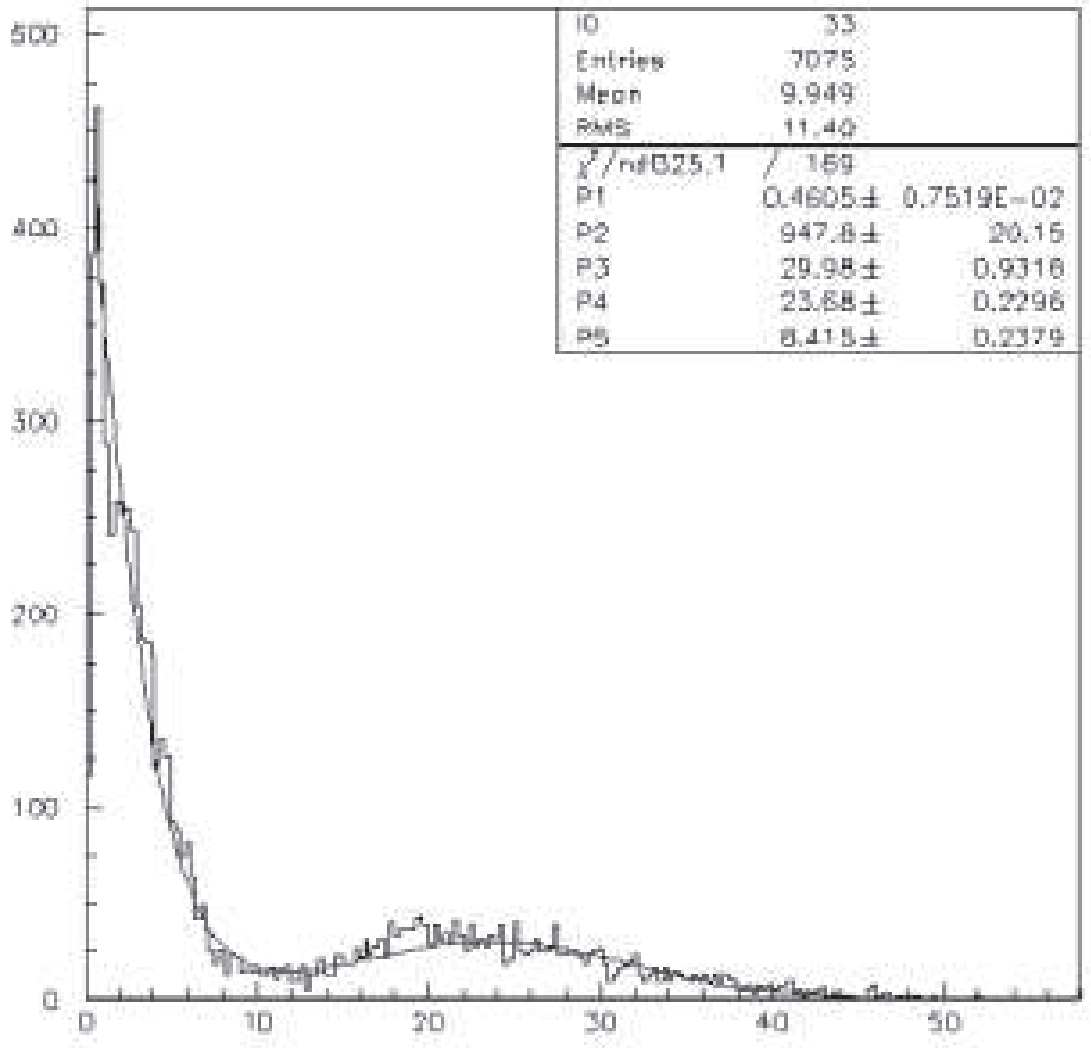


Figura 3.8: Histograma e ajuste da fdp para o tom de freqüência de índice 12 da classe E. O ajuste é rejeitado pelo teste de aderência do chi-quadrado, uma vez que o valor do χ^2 obtido é igual a 325,7 (para 159 graus de liberdade), superior ao valor limite do χ^2, o qual é igual a 189,4.

ses, potencialmente, apresentarão uma maior tendência a erros de classificação. A partir deste ponto, nesta e nas demais análises que serão apresentadas com base em distância, passaremos a empregar como métrica o quadrado da distância euclidiana, ao invés da distância euclidiana. Como a medida de distância entre espectros e curvas foi definida conforme a Equação 2.5 (função de distância empírica), a qual emprega o quadrado da distância euclidiana, se todas as demais medidas de distância, mesmo aquelas que não envolvem curvas (como neste caso, onde se apresentam as distâncias entre os centros das classes), empregarem a mesma métrica, será possível compará-las diretamente. Na Tabela 3.4 são apresentados os valores das distâncias

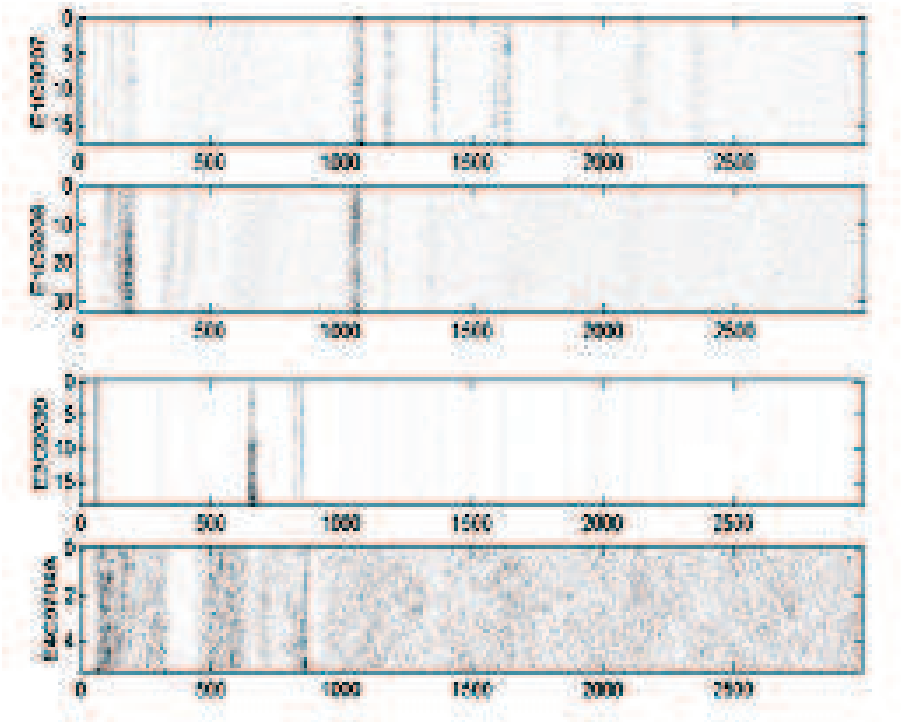


Figura 3.9: Exemplos de espectrogramas extraídos a partir de corridas de navios pertencentes à classe E.

entre os centros das classes. A distância euclidiana entre os centros de duas classes arbitrárias A e B, de centros

$$\overline{x}_A = \overline{x}_{1A}, \overline{x}_{2A}, \cdots, \overline{x}_{dA} \tag{3.2}$$

e

$$\overline{x}_B = \overline{x}_{1B}, \overline{x}_{2B}, \cdots, \overline{x}_{dB}, \tag{3.3}$$

onde d é a dimensão dos dados (neste caso, $d = 557$) é obtida como

$$d = ((\overline{x}_{1A} - \overline{x}_{1B})^2 + (\overline{x}_{2A} - \overline{x}_{2B})^2 + \cdots + (\overline{x}_{dA} - \overline{x}_{dB})^2)^{1/2} \tag{3.4}$$

Em negrito, temos os menores valores observados para cada linha ou classe, em unidade arbitrária (u.a.). Analisando a Tabela 3.4, verificamos que as classes A e C (174,2) e B e H (161,3) apresentam centros mais próximos dentre todas as demais classes. Em seguida, temos a classe C e a classe H (272,3). Baseado nessas observações, é possível supor que, do ponto de vista da classificação, essas classes tendem a apresentar maior grau de confusão entre si.

Tabela 3.4: Distâncias euclidianas entre os centros das classes (u.a.).

	Classe A	***Classe B***	***Classe C***	***Classe D***	***Classe E***	***Classe F***	***Classe G***	***Classe H***
Classe A	0	789,6	**174,2**	1011,2	1049,8	533,6	1156,0	492,8
Classe B	789,6	0	404,0	1030,4	882,1	697,0	529,0	**161,3**
Classe C	**174,2**	404,00	0	1122,3	665,6	510,8	967,2	272,3
Classe D	1011,2	1030,4	1122,3	0	**640,1**	1317,7	2590,8	900,0
Classe E	1049,8	882,1	665,6	**640,1**	0	1560,3	3025,0	894,0
Classe F	533,6	697,0	510,8	1317,7	1560,3	0	1383,8	**342,3**
Classe G	1156,0	**529,0**	967,2	2590,8	3025,0	1383,8	0	571,2
Classe H	492,8	**161,3**	272,3	900,0	894,0	342,3	571,2	0

O gráfico da Figura 3.10 mostra a distância entre os centros das classes. Na escala vertical, o valor d representa as distâncias euclidianas entre os centros. É possível notar que os centros de algumas classes apresentam-se mais distanciados entre si que os demais. Isto ocorre, por exemplo, entre as classes D e G e E e G, para as quais espera-se que a confusão entre seus espectros seja menor que entre os espectros das outras classes.

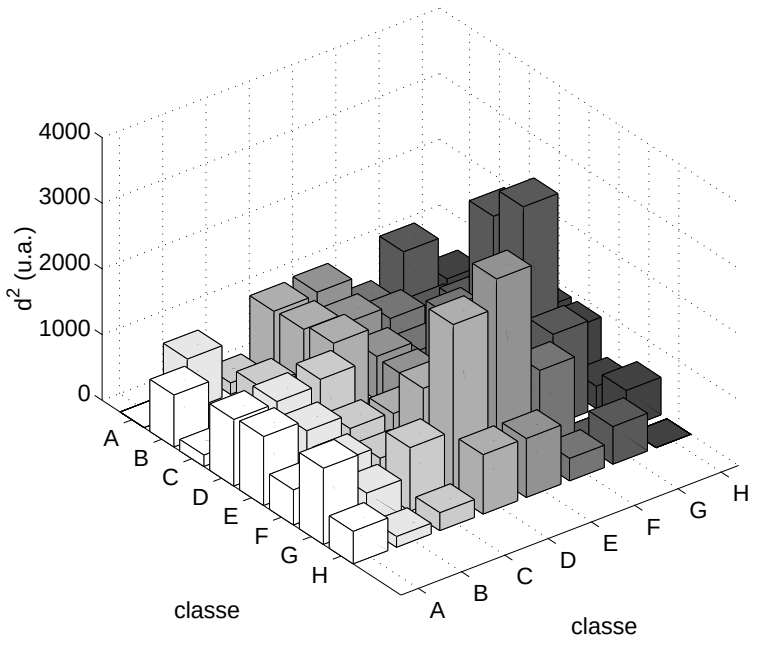


Figura 3.10: Médias das distâncias euclidianas entre centros.

3.2.3 Distâncias entre os espectros das classes

A análise dos valores médios das distâncias entre os espectros de uma classe e aqueles pertencentes às demais classes permite verificar quão bem definidos e separados estão os agrupamentos associados a cada classe. Se os espectros das classes estiverem agrupados distantes uns dos outros, as menores distâncias ocorrerão entre os espectros da mesma classe e não entre espectros de classes diferentes. Na Tabela 3.5, são apresentadas as distâncias envolvidas nesta análise. É possível observar que os espectros pertencentes às classes B e H encontram-se a uma distância menor de espectros de outras classes que de espectros pertencentes à própria classe. Considerando a classe B, os seus espectros se encontram mais próximos, em média, dos espectros da classe G (235,28) do que dos espectros pertencentes à sua classe (242,67). O mesmo é verificado para a classe H com relação à classe G (238,19). Também é possível observar que os valores das distâncias estão bastante próximos entre si, indicando uma pequena separação entre as diferentes classes. Considerando, por exemplo, a classe A, verificamos que a diferença percentual da distância média dos espectros da própria classe (219,53) em relação àqueles da classe de maior distância, a classe B (258,90), é de apenas 18%.

Tabela 3.5: Média das distâncias euclidianas entre espectros (u.a.).

	Classe A	***Classe B***	***Classe C***	***Classe D***	***Classe E***	***Classe F***	***Classe G***	***Classe H***
Classe A	**219,5**	258,9	242,3	240,4	257,8	252,4	234,7	254,9
Classe B	258,9	242,7	260,8	252,3	266,8	267,3	**235,3**	257,06
Classe C	242,3	260,8	**239,1**	251,9	261,1	261,6	241,7	259,2
Classe D	240,4	252,3	251,9	**198,3**	240,1	255,0	241,0	252,2
Classe E	257,8	266,8	261,1	240,1	**231,8**	275,0	262,0	268,9
Classe F	252,4	267,3	261,8	255,0	275,0	**239,8**	248,1	261,3
Classe G	234,7	235,3	241,7	241,0	262,0	248,1	**182,6**	238,2
Classe H	254,9	257,1	259,2	252,2	268,9	261,3	**238,2**	246,6

O gráfico da Figura 3.11 apresenta as médias das distâncias entre os espectros das classes. Foi utilizado, na escala vertical, o valor $280 - d^2$, onde d^2 representa a média do quadrado das distâncias euclidianas entre os espectros, assim, as menores distâncias correspondem aos maiores valores.

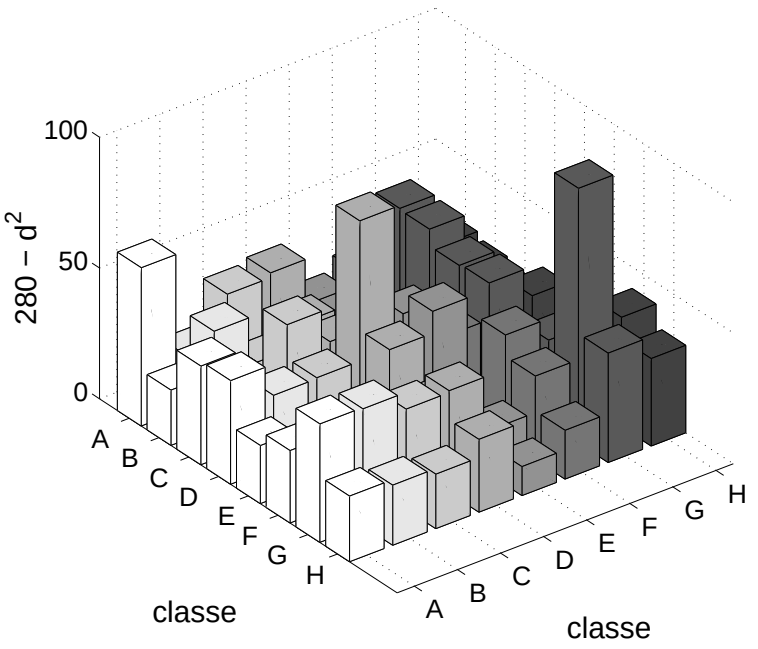


Figura 3.11: Médias das distâncias entre espectros.

3.3 Implementação

Inicialmente, foi implementado um classificador baseado nas distâncias dos sinais aos centros das classes. Adotou-se, como critério de classificação, atribuir o espectro à classe cuja distância euclidiana ao seu centro é a menor dentre todas as demais. Para este classificador, uma eficiência média de 77,60% foi obtida, insatisfatória para a aplicação. Iniciou-se, então, a investigação de um classificador baseado em curvas principais.

3.3.1 O classificador baseado em curvas principais

A abordagem empregada na construção do classificador é baseada no trabalho publicado por Chang e Ghosh (1998) [34]. Para o projeto do classificador proposto, procedeu-se à extração das curvas principais de cada classe, utilizando os espectros contidos nos conjuntos de projeto. Adotou-se como critério de classificação atribuir o espectro à classe cuja distância euclidiana a sua curva representativa é a menor dentre todas as demais. Este classificador pressupõe, portanto, que cada classe possa ser bem caracterizada por uma dada curva, assim como supõe que a distância entre

cada espectro e a curva de sua classe, para a maior parte dos espectros, é menor que a distância deste espectro específico às demais curvas.

Considerando que um espectro é um ponto num espaço multidimensional (557 componentes) e que a curva é composta por segmentos de reta, a distância entre um dado espectro e a curva é definida conforme a Equação 2.5. A Figura 3.12 ilustra o processo de classificação, considerando que os dados são representados num espaço com apenas duas dimensões (no plano cartesiano). Nela são mostrados dois pontos no plano, e_1 e e_2, representando dois eventos em $\Re^2$. Também são mostradas duas curvas principais representativas das classes 1 e 2, $\mathbf{f}_1(t)$ e $\mathbf{f}_2(t)$. A fim de classificar o evento e_1 mede-se a distância euclidiana deste evento aos segmentos de cada uma das curvas principais representativas das classes envolvidas. As distâncias do evento 1 às curvas 1 e 2 estão representadas na figura como d_{11} e d_{12}. Como e_1 está mais próximo da curva representativa da classe 1, $\mathbf{f}_1(t)$, do que da curva representativa da classe 2, $\mathbf{f}_2(t)$, classifica-se e_1 como pertencente à classe 1. Da mesma forma, as distâncias do evento 2 e_2 às curvas 1 e 2 são representadas, respectivamente, por d_{11} e d_{12}. Este evento deve ser classificado como pertencente à classe 2, visto que ele está mais próximo da curva $\mathbf{f}_2(t)$. Num espaço de dimensão elevada, como o abordado na aplicação em pauta, o processo de classificação é realizado de forma similar. Cada evento constitui-se num ponto, cujas distâncias euclidianas às curvas representativas das classes determinam sua classificação.

Por questão de simplicidade, o termo distância euclidiana continuará a ser empregado neste texto. No entanto, a partir deste ponto, ele se refere ao quadrado da distância euclidiana entre o espectro e seu ponto de projeção sobre a curva principal, de acordo com a definição de função de distância (veja a Equação 2.3).

Na seção 3.4, serão apresentados os resultados de classificação obtidos com o emprego das curvas principais extraídas.

3.3.2 O algoritmo de k-segmentos não-suave

Na extração das curvas utilizou-se o algoritmo de k-segmentos não-suave, descrito na seção 2.2.4. O algoritmo sofreu três pequenas modificações. A primeira modificação foi necessária à aplicação das curvas principais como ferramentas volta-

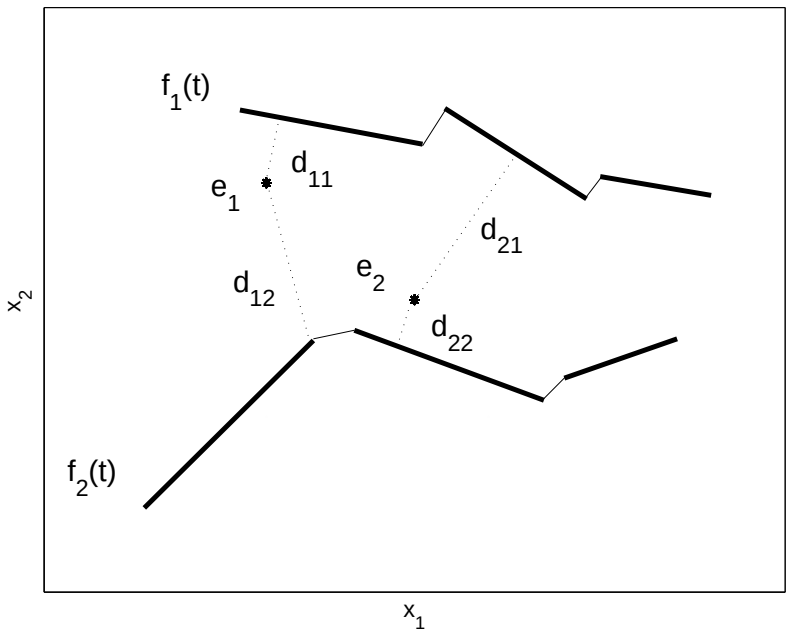


Figura 3.12: Classificação de um evento.

das para a tarefa de classificação, a segunda visou otimizar a alocação de memória durante o processamento e a última procurou ampliar a robustez do algoritmo.

Inicialmente, o algoritmo de k-segmentos não-suave remove a média das variáveis aleatórias pertencentes ao conjunto de dados. O efeito prático é o de trazer o centro da curva principal para o origem do espaço de coordenadas. Se empregado desta maneira, o algoritmo de k-segmentos não-suave extrairia cada curva com forma própria, definida a partir do conjunto de dados de projeto correspondente, mas todas as curvas compartilhariam o mesmo centro (a origem), o que dificultaria muito a tarefa de classificação que, neste trabalho, se baseia na comparação das distâncias do evento às curvas. Assim, é importante manter a separação entre os centros das classes, resultado da distribuição natural dos dados e do pré-processamento, o qual, neste tipo de aplicação, objetiva ressaltar as diferenças entre as classes, mediante a extração de informação discriminante. Eliminou-se, então, este passo, comentando-se a segunda linha de código, $X = X - repmat(mean(X), size(X, 1), 1)$.

Foi necessário alterar a rotina map_to_arcl.m para que, ao invés de processar todos os espectros pertencentes ao conjunto de dados, um espectro fosse processado a cada vez. Esta alteração foi realizada porque a dimensão do espaço de definição dos dados de entrada é muito elevada, demandando a alocação de uma grande quan-

tidade de memória. A rotina modificada, map_to_arcl_mod.m, consta das rotinas listadas no Anexo A.

A última modificação foi feita na linha 38 do código da rotina construct_hp.m, $mind = mind + eye(size(mind, 1)) * 10000$. O valor 10000 foi substituído por *realmax* (máximo valor real representável pelo MatLab) o que garante que, na maioria dos casos práticos, os valores da diagonal principal da matriz *mind* assumirão uma faixa de valores muito maior que a inicial, conforme esperado.

Na extração das curvas foram empregados os conjuntos de projeto abordados na Seção 3.2 e os seguintes valores de parâmetros de entrada [5]: $k\_max = 40$, $alpha = 1$ e $lambda = 1$.

3.3.3 Extração das curvas principais

Inicialmente, será avaliado o impacto do número de segmentos que compõem a curva principal de cada classe e da metodologia de tratamento dos espectros de entrada na eficiência de generalização do classificador. Para realizar esta tarefa foi consirado um número máximo crescente de segmentos a extrair, desenvolvendo classificadores para dados submetidos a diferentes metodologias de tratamento, entre elas: sem normalização (método AN1), extraindo a média e normalizando cada variável pelo seu desvio padrão (método AN2) e, finalmente, pela média de dois espectros resultantes do método AN2 (método AN3). A normalização dos espectros visa a ajustar a faixa dinâmica dos dados, permitindo que curvas de comprimentos mais uniformes sejam estraídas. A motivação do emprego de médias foi introduzida na Seção 3.2, com base no contido em [49], e visa a redução do ruído de fundo.

Os resultados são apresentados na Tabela 3.6. Como podemos observar, há um aumento da eficiência de classificação com o aumento do número de segmentos por curva e com a adoção de uma metodologia de normalização. Além disso, nota-se que a utilização da média de dois espectros normalizados (método AN3) tende a produzir melhores resultados. Foi verificado, ainda, que o ganho de eficiência tende a se reduzir com o aumento do número de segmentos, dando indícios de que a curva de eficiência tende, assintóticamente, a um valor máximo, conforme esperado. As

[5] A forma de chamar a rotina de k-segmentos não-suave e as descrições das variáveis e dos parâmetros de entrada e saída estão contidas na Seção 2.2.4.

piores eficiências de classificação são verificadas para a classe C, fato coerente com os resultados iniciais de distância entre as classes e com a análise estatística realizada. Com relação aos resultados obtidos para as classes A, B e E, apontadas como preocupantes, juntamente com a classe C, na análise estatística preliminar realizada, verifica-se que as classes A e B apresentam resultados relativamente bons, inclusive para curvas com apenas um segmento[6]. A classe E, no entanto, exibe resultados relativamente ruins considerando-se uma curva de apenas um segmento mas, já para uma curva com cinco segmentos os resultados são relativamente bons, indicando que a curva principal é capaz de representar bem esses dados, permitindo que se obtenham bons resultados de classificação. Como nesta análise foram empregados os conjuntos de projeto, por enquanto, nada se pode inferir com relação à capacidade de generalização do classificador baseado em curvas principais.

Considerando as curvas com dez segmentos e a normalização AN3, a pior eficiência de detecção obtida foi de 88,8%, atingindo uma eficiência média de 95,7% para o conjunto de projeto, o que é um resultado bom, conforme será mostrado na Seção 3.4, onde são comparados os resultados obtidos neste trabalho e em outros, que empregaram os mesmos dados.

Tabela 3.6: Eficiências de detecção das classes em função do número de segmentos e do pré-processamento adotado. Resultados obtidos para dados de projeto em porcentagem.

Nr.Max.Seg. e Análise Adotada		***Classe A***	***Classe B***	***Classe C***	***Classe D***	***Classe E***	***Classe F***	***Classe G***	***Classe H***	***Média***
1 seg.	**AN1**	87,5	80,0	56,6	93,5	42,9	93,5	84,9	87,2	**78,2**
	AN2	88,7	78,0	59,8	90,8	70,2	92,2	89,9	80,5	**81,3**
	AN3	92,6	82,4	63,9	92,8	72,8	96,2	95,2	84,4	**85,0**
5 seg.	**AN1**	94,3	80,1	74,2	90,7	84,0	89,3	92,5	86,8	**86,5**
	AN2	90,3	91,1	80,9	90,8	94,2	95,7	91,7	84,1	**89,8**
	AN3	98,0	96,6	77,0	93,5	94,7	98,5	96,4	88,2	**92,9**
10 seg.	**AN1**	96,5	88,9	82,6	90,8	91,5	95,3	91,8	85,2	**90,3**
	AN2	91,8	93,2	85,4	91,5	95,8	96,1	91,8	84,7	**91,3**
	AN3	97,4	95,2	88,8	94,3	96,3	98,6	98,2	96,5	**95,7**

Procedemos, em seguida, à extração do número máximo de segmentos possível para cada curva, dado que o algoritmo de k-segmentos não-suave promove a inserção

[6]Cada curva representativa de uma dada classe, neste caso, trata-se de apenas um segmento, o qual possui a direção da primeira componente principal dos dados da classe.

construtiva de segmentos. Para esta extração, foram considerados apenas os dados processados de acordo com as metodologias AN2 e AN3, visto que estas apresentaram os resultados mais promissores.

A medida que novos segmentos são inseridos, durante a fase de extração de uma curva principal, a função de distância empírica (Equação 2.4), associada à curva e aos dados, deve decrescer continuamente. Assim, o comportamento da função de distância empírica, cujos valores são diponibilizados pelo algoritmo de k-segmentos não-suave, conforme descrito no Item 2.2.4, é um indicativo importante da correção do processo de extração de uma curva principal. A Figura 3.13 mostra os valores assumidos pela função de distância empírica, para cada uma das curvas representativas das classes, após cada adição de um novo segmento, considerando dados de projeto obtidos a partir da metodologia AN3. Em cada gráfico, duas curvas foram incluídas, cada uma se referindo ao conjunto de dados de uma determinada classe. Observe que os valores da função de distância empírica são sempre decrescentes, conforme esperado, indicando que, a medida que novos segmentos são inseridos, a curva fica mais ajustada aos dados. Comportamento semelhante também foi observado para a extração das curvas principais empregando dados de projeto obtidos a partir da metodologia AN2.

Antes de prosseguir com a apresentação dos resultados finais de classificação, serão exploradas algumas análises adicionais que visam caracterizar o problema da classificação baseado nas curvas finais extraídas. Inicialmente, analisaremos as curvas, apresentando a quantidade de segmentos que compõem cada uma delas e seus comprimentos. A seguir, apresentaremos as médias das distâncias entre os espectros de cada classe e as curvas principais finais, relacionando esses valores com os obtidos para as distâncias médias entre espectros. Por fim, serão apresentadas algumas medidas de distâncias entre as curvas principais extraídas.

3.3.4 Quantidade de segmentos e comprimentos das curvas

Através da análise dos comprimentos das curvas, é possível ter-se uma idéia da complexidade das estruturas dos dados e da sua distribução no espaço de coordenadas. Curvas mais longas podem indicar a presença de classes compostas por navios que produzem informação bastante diversificada, fazendo com que seus es-

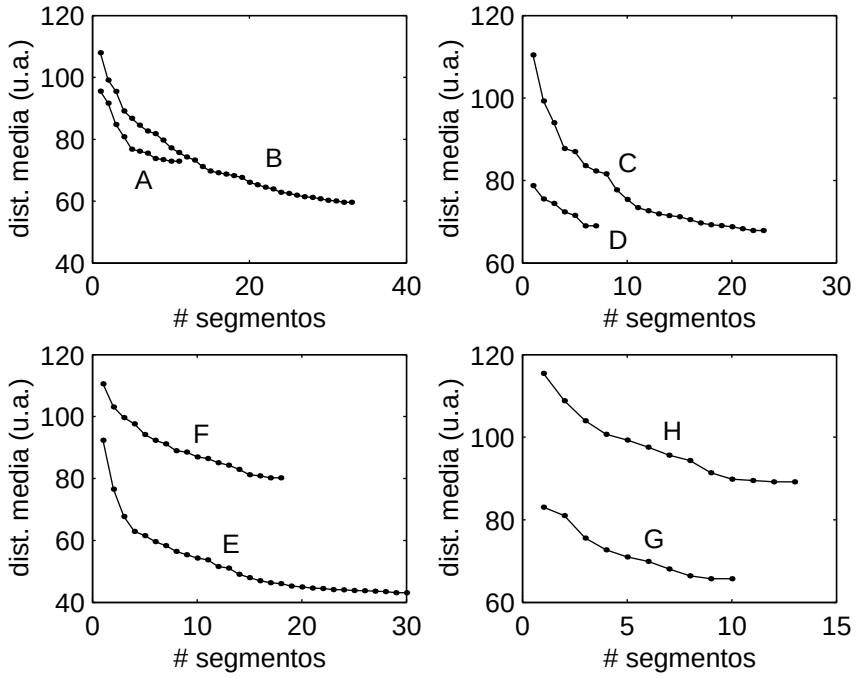


Figura 3.13: Valores da função de distância empírica em função do número de segmentos das curvas de cada classe. A classe é indicada pelas letras junto às curvas.

pectros formem agrupamentos distantes uns dos outros. Por outro lado, curvas mais curtas podem indicar que os dados pertencem a uma classe mais homogênea, compostas por um ou mais grupamentos relativamente próximos.

O número máximo de segmentos extraídos para as curvas de cada classe e seus comprimentos são apresentados nas Tabelas 3.7 e 3.8. Pode ser observado que, nos dois casos, o comprimento das curvas é diretamente proporcional ao número de segmentos. Como se espera que ocorra uma redução do ruído de fundo para dados da normalização AN3, com relação aos dados da normalização AN2, inicialmente, se supos que os primeiros estariam menos dispersos e, em conseqüência produziriam curvas mais compactas, ou seja, mais curtas e formadas por um número menor de segmentos. No entanto, o inverso foi verificado: as curvas extraídas para dados da normalização AN3 possuem um número maior de segmentos e são mais longas que as curvas ajustadas aos dados da normalização AN2. Uma explicação intuitiva atribuída a este fato é que, desconsiderando-se o ruído de fundo, os tons irradiados por cada classe deveriam formar agrupamentos, de acordo com a condição operativa assumida pelo navio, no momento da sua passagem pela raia acústica, e das características peculiares de cada sistema de propulsão. Ao se adicionar o ruído de fundo,

a distribuição dos sinais resultantes torna-se mais uniforme e aglomeramentos, antes bem definidos, unem-se. Assim, suspeita-se que, para um nível de ruído maior, como no caso da normalização AN2, com relação à normalização AN3, a quantidade de grupamentos seja menor, acarretando em uma curva composta por um número menor de segmentos.

Da análise dos resultados contidos na Tabela 3.6, tivemos indícios que a eficiência de classificação varia diretamente com o número de segmentos das curvas principais. Supuzemos, também que, para um mesmo número de segmentos, o classificador empregando curvas, obtidas a partir dos dados tratados conforme o método AN3 apresenta eficiências de classificação mais elevadas que o classificador baseado em curvas obtidas a partir dos dados obtidos segundo a metodologia AN2. Supõem-se, portanto, com base nessas duas evidências, que as eficiências de classificação finais para o classificador baseado nas curvas principais extraídas a partir dos dados processados segundo o método AN3 serão melhores que as obtidos empregando as curvas extraídas a partir dos dados processados conforme o método AN2.

Tabela 3.7: Número máximo de segmentos e comprimentos das curvas de cada classe (u.a.), empregando a normalização AN2.

	Classe A	***Classe B***	***Classe C***	***Classe D***	***Classe E***	***Classe F***	***Classe G***	***Classe H***
Nr. Seg.	7	23	16	2	27	10	7	5
Comp.	109,4	512,2	315,1	26,4	471,7	202,0	113,5	82,6

Tabela 3.8: Número máximo de segmentos e comprimentos das curvas de cada classe (u.a.), empregando a normalização AN3.

	Classe A	***Classe B***	***Classe C***	***Classe D***	***Classe E***	***Classe F***	***Classe G***	***Classe H***
Nr. Seg.	10	33	22	6	29	17	9	12
Comp.	173,6	718,6	479,6	73,9	533,5	341,8	158,3	228,7

Todos os resultados que serão apresentados a partir deste ponto se referem às curvas principais finais, aquelas cuja quantidade de segmentos é maxima, conforme contido nas Tabelas 3.7 e 3.8.

3.3.5 Distâncias entre espectros e curvas

A Tabela 3.9 mostra os valores médios das distâncias entre os espectros de cada classe e as curvas. Pode ser observado que os valores contidos na diagonal, os quais representam as distâncias médias entre os espectros de uma classe e a curva representativa da classe, são menores que os demais, conforme esperado, indicando que a classificação baseada em curvas deve apresentar bons resultados. A Figura 3.14 mostra, graficamente, os resultados obtidos. Novamente foi utilizado, na escala vertical, o valor $280 - d^2$, onde d^2 representa a média do quadrado das distâncias euclidianas entre os espectros evidenciando, assim, as menores distâncias, as quais aparecem na diagonal.

Tabela 3.9: Média das distâncias entre espectros e curvas (u.a.).

	Classe A	***Classe B***	***Classe C***	***Classe D***	***Classe E***	***Classe F***	***Classe G***	***Classe H***
Classe A	**75,4**	146,1	114,6	130,5	130,2	128,1	135,8	123,3
Classe B	149,2	**65,8**	134,4	135,0	141,2	144,3	134,2	121,4
Classe C	129,2	143,7	**72,8**	139,9	127,4	139,4	241,7	259,2
Classe D	132,0	130,4	111,4	**70,6**	110,8	131,5	135,0	113,0
Classe E	146,6	142,4	109,3	129,4	**47,6**	154,0	138,4	133,4
Classe F	137,4	158,2	134,9	141,3	140,8	**84,8**	145,9	137,8
Classe G	121,6	113,6	110,7	115,5	126,7	126,1	**65,0**	99,5
Classe H	142,8	139,3	133,3	138,6	144,6	141,2	140,7	**92,5**

Analisando os valores das distâncias entre espectros (Tabela 3.5) pudemos concluir que os dados referentes às diversas classes não estão contidos em agrupamentos distantes entre si (visto que, por exemplo, os espectros da classe B estão mais distantes entre si que dos espectros da classe G). Quando as curvas principais foram empregadas como ferramentas de representação e, ao invés de medir a distância entre os espectros, mediu-se a distância dos espectros às curvas, foi possível verificar que, aparentemente, as medidas de distância entre espectros e curvas podem discriminar, eficientemente, espectros de uma classe arbitrária. Isto ocorre porque, conforme definido por HS [1], as curvas passam no meio dos dados, fornecendo uma representação espacial conveniente destes.

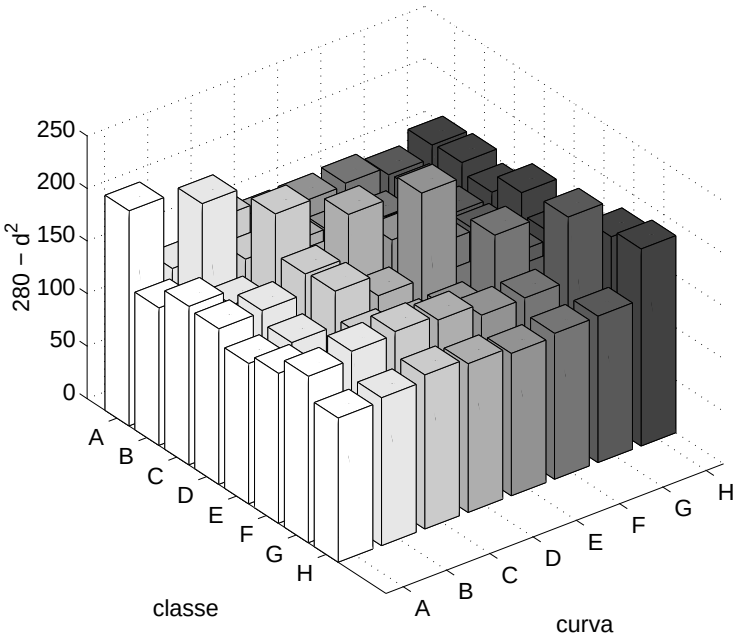


Figura 3.14: Médias das distâncias entre espectros e curvas (unidade arbitrária).

3.3.6 Distâncias entre curvas

companhia

A fim de avaliar a complexidade do processo de classificação, buscamos avaliar as distâncias entre as curvas principais de cada classe, dado que o critério de classificação adotado foi baseado na distância de um dado espectro recebido à curva representativa de cada classe. A fim de calcular os valores de distâncias entre curvas, empregou-se um método aproximado, baseado na distância entre pontos pertecentes às curvas. Inicialmente, foi obtido, para cada curva, um conjunto de pontos pertencentes a elas. A densidade de pontos foi mantida constante ao longo da curva, ou seja, a quantidade de pontos obtidos, para cada curva, é diretamente proporcional ao seu comprimento. Estabeleceu-se que, a cada unidade arbitrária de comprimento de uma curva, seriam determinados dois pontos. Assim, por exemplo, no caso da curva da classe A, para dados obtidos para a normalização AN2, a qual possui um comprimento igual a 109,4 u.a., foram determinadas as coordenadas de 219 pontos, sobre essa curva.

A escolha de uma métrica adequada à medida da distância entre curvas não é uma tarefa simples. A métrica mais básica e intuitiva é a distância mínima entre

elas. A distância mínima entre a curva A e a B pode ser definida como a menor distância entre os pontos mais próximos de cada uma das curvas. Formalmente, temos:

$$d_{min}(A,B) = \inf_{a \in A}\{\inf_{b \in B}\{d(a,b)\}\} \tag{3.5}$$

onde a e b são pontos pertencentes às curvas A e B, respectivamente, e $d(a,b)$ é a distância entre os pontos a e b. Neste trabalho, o quadrado da distância euclidiana foi adotado como métrica. A Tabela 3.10 apresenta os resultados das distâncias mínimas entre as curvas principais das classes, duas a duas.

Tabela 3.10: Distâncias mínimas entre curvas (u.a.).

	Classe A	***Classe B***	***Classe C***	***Classe D***	***Classe E***	***Classe F***	***Classe G***	***Classe H***
Classe A	0	49,1	**14,4**	26,7	27,5	20,6	40,0	16,7
Classe B	49,1	0	38,5	26,7	38,7	38,6	22,3	**11,1**
Classe C	**14,4**	38,5	0	20,1	13,8	23,7	31,7	23,7
Classe D	26,7	26,7	20,1	0	24,2	25,3	20,7	**17,7**
Classe E	27,5	38,7	**13,8**	24,2	0	17,4	27,9	26,8
Classe F	20,6	38,6	23,7	25,3	**17,7**	0	27,2	21,0
Classe G	40,0	22,3	31,7	20,7	27,9	27,2	0	**15,7**
Classe H	16,7	**11,1**	23,7	17,7	26,8	21,0	15,7	0

Analisando os resultados mostrados na Tabela 3.10, nota-se que a distância entre as curvas das classes A e C é a menor dentre as distâncias mínimas dentre as distâncias da curva da classe A às demais, o que indica uma possível confusão na classificação dos espectros dessas classes. O mesmo ocorre, por exemplo, para as classes B e G com relação à classe H.

A distância mínima não considera o posicionamento relativo entre as curvas. Por este motivo, considerou-se a distância de *Hausdorff* não-generalizada [76], comumente empregada no reconhecimento de objetos ou padrões em imagens. A distância de *Hausdorff* é formalmente definida como:

$$h(A,B) = \sup_{a \in A}\{\inf_{b \in B}\{d(a,b)\}\}. \tag{3.6}$$

A distância de *Hausdorff* da curva A para a curva B pode ser entendida como

a menor distância que separa A de B, no pior caso. Ela caracteriza melhor a noção da distância entre curvas porque, ao contrário da distância mínima, que considera apenas a distância entre os pontos mais próximos de cada curva, ela considera a totalidade dos pontos de cada uma, dando uma noção do posicionamento relativo entre elas. A Tabela 3.11 apresenta a distância de Hausdorff entre cada curva. Observe que a distância de *Hausdorff* não é simétrica, ou seja, geralmente, $h(A, B) \neq h(B, A)$, não sendo uma métrica.

Tabela 3.11: Distâncias de *Hausdorff* entre curvas (u.a.).

	Classe A	***Classe B***	***Classe C***	***Classe D***	***Classe E***	***Classe F***	***Classe G***	***Classe H***
Classe A	0	162,3	147,8	157,1	157,8	150,6	161,3	**137,8**
Classe B	192,8	0	170,7	210,0	176,1	187,9	185,1	**152,6**
Classe C	213,0	241,6	0	213,9	243,7	**207,7**	250,8	221,2
Classe D	142,2	**115,3**	128,7	0	130,2	152,7	119,6	152,2
Classe E	211,7	206,6	**206,0**	218,5	0	211,5	228,2	200,6
Classe F	150,9	181,2	**149,5**	166,1	155,8	0	166,3	162,1
Classe G	163,1	140,3	126,6	**118,8**	163,8	177,3	0	177,0
Classe H	164,1	134,8	155,1	164,0	161,0	159,0	166,7	0

Os valores das distâncias de Hausdorff indicam que as curvas das classes D e B estão relativamente mais próximas entre si que as demais. Esta proximidade não foi detectada através da medida da distâncias mínimas entre curvas. Os centros dessas duas classes não estão dentre os mais próximos, ao contrário, com relação aos demais, eles estão distantes entre si; o mesmo ocorrendo com relação à distância entre os espectros das duas classes. Muito embora não se possa justificar a proximidade relativa entre essas curvas com base em resultados anteriores, é importante que os resultados obtidos sejam empregados na análise das matrizes de confusão dos classificadores, a serem apresentadas na Seção 3.4. Nota-se, também, que os valores de distâncias obtidos não estão fortemente correlacionadas com as distâncias mínimas. Enquanto que, em termos da distância mínima, a classe C é a classe mais próxima da classe A, considerando-se a distância de Hausdorff, a classe A tem a classe H como a mais próxima.

Como a distância de Hausdorff considera a maior dentre as distâncias verificadas entre os pontos pertencentes à curva de origem e à curva alvo, ela é muito influenciada pela presença de intrusos (outliers). Ou seja, um pequeno grupamento

de espectros pertencente ao conjunto de projeto, distante da maioria dos dados, pode definir um segmento, relativamente afastado dos demais, o qual, por si só, determinaria a distância em questão. Na busca por uma medida robusta que pudesse avaliar, convenientemente, a separação relativa entre curvas, decidiu-se obter a média das distâncias mínimas entre as curvas. Com o objetivo de privilegiar segmentos mais representativos em cada curva (segmentos associados a um maior número de espectros), as curvas foram mapeadas novamente, desta vez, empregando-se uma densidade de pontos não-uniforme, definida em função do número de eventos do conjunto de projeto associado a cada segmento. Ao invés de empregar um número de pontos total igual ao dobro do comprimento das curvas, para cada segmento foram calculadas as coordenadas de n pontos pertencentes ao segmento, onde n foi arbitrado como a décima parte da quantidade de pontos associados a cada segmento. Considere, por exemplo, uma curva definida por dois segmentos, extraída a partir de um conjunto de projeto constituído por 100 pontos, estando o primeiro segmento associado a 60 pontos e o segundo a 40. Neste caso, seriam obtidos 10 pontos sobre a curva, 6 pertencentes ao primeiro segmento e 4 ao segundo. Essa nova medida foi denominada média ponderada das distâncias mínimas. A Tabela 3.12 contém os valores obtidos nesta análise. Ao contrário da distância mínima entre curvas, a média ponderada das distâncias entre curvas não é, necessariamente, simétrica.

Tabela 3.12: Média ponderada das distâncias mínimas entre curvas (u.a.).

	Classe A	***Classe B***	***Classe C***	***Classe D***	***Classe E***	***Classe F***	***Classe G***	***Classe H***
Classe A	0	75,9	**43,0**	57,2	57,4	56,4	62,8	52,4
Classe B	92,2	0	81,8	76,9	85,3	87,3	78,8	**66,8**
Classe C	63,8	78,3	0	70,9	**60,4**	72,4	72,4	66,1
Classe D	61,9	62,0	42,2	0	**41,8**	61,5	73,5	51,7
Classe E	103,2	98,6	**65,4**	85,9	0	110,5	99,0	95,8
Classe F	58,4	81,4	**56,1**	61,2	61,2	0	66,1	58,1
Classe G	63,3	53,9	52,6	52,6	64,0	64,0	0	**40,3**
Classe H	54,3	52,1	**47,7**	47,8	55,2	53,6	51,4	0

Da análise dos resultados contidos na Tabela 3.12, verifica-se, por exemplo, que a curva da classe C é a mais próxima da classe A, assim como a classe H é a mais próxima da classe B, indicando uma possível confusão entre os espectros dessas classes. Comparando-se os valores obtidos nessa análise com aqueles obtidos a partir

das distâncias mínimas entre as curvas, verifica-se que os dados estão correlacionados. As duas análises coincidem ao indicar a curva da classe C como a mais próxima das classes A e B. O mesmo ocorre com relação à classe H e as classes B e G.

3.4 Resultados de classificação

De posse das curvas finais[7], procedemos à avaliação da matrizes de confusão para o conjuntos de projeto e de teste, processados segundo as metodologias AN2 e AN3, cujos resultados estão contidos nas Tabelas 3.13, 3.14, 3.15 e 3.16. Nestas matrizes, cada linha representa os espectros restritos a uma dada classe, que são apresentados na entrada do classificador, e cada coluna responde pelos espectros que são classificados como pertencentes à própria classe (diagonal da matriz) e às demais classes. A matriz de confusão permite, portanto, verificar a eficiência de detecção de cada classe, assim como identificar em relação a quais classes um maior número de erros de detecção é cometido. A comparação dos resultados obtidos com os conjuntos de projeto e teste permite avaliar a capacidade de generalização do classificador.

Tabela 3.13: Matriz de confusão para o melhor classificador obtido pela metodologia AN2. Resultados para dados de *projeto* em porcentagem.

	Classe A	***Classe B***	***Classe C***	***Classe D***	***Classe E***	***Classe F***	***Classe G***	***Classe H***	***Total de Eventos***
Classe A	**93,7**	0,3	4,9	0	0,5	0,4	0,1	0,1	810
Classe B	0,1	**97,9**	0,4	0	0,1	0	1,0	0,5	858
Classe C	3,4	0,3	**95,2**	0,1	0,2	0	0,6	0,2	959
Classe D	0,5	0,7	2,3	**91,2**	0,2	1,2	2,2	1,7	1024
Classe E	0,2	0,1	0,2	0	**99,3**	0	0,1	0,1	1010
Classe F	0,7	0,1	1,0	0,2	0,2	**97,1**	0,2	0,5	978
Classe G	0,4	1,6	1,1	1,1	0,4	0,3	**93,9**	1,2	1071
Classe H	1,1	2,8	2,6	1,2	0,6	0,3	2,6	**88,8**	848
Eficiência Média = 94,6									

Os resultados finais são bastante satisfatórios, tendo-se obtido uma eficiência média igual a 94,6% e 92,4%, respectivamente, para os conjuntos de projeto e teste

[7]As curvas principais finais foram extraídas para um número máximo de segmentos, conforme contido nas Tabelas 3.7 e 3.8.

Tabela 3.14: Matriz de confusão para o melhor classificador obtido pela metodologia AN2. Resultados para dados de *teste* em porcentagem.

	Classe A	***Classe B***	***Classe C***	***Classe D***	***Classe E***	***Classe F***	***Classe G***	***Classe H***	***Total de Eventos***
Classe A	**90,5**	0,3	6,9	0,5	0,8	0,4	0,2	0,4	1620
Classe B	0,1	**96,0**	0,4	0,1	0,2	0,1	2,6	0,5	2574
Classe C	4,8	0,7	**90,7**	0,7	0,5	0,4	0,9	1,3	3836
Classe D	0,6	1,1	2,0	**91,2**	0,1	1,2	2,5	1,4	2048
Classe E	0,4	0,2	0,8	0,1	**97,6**	0,1	0,4	0,4	6060
Classe F	0,7	0,1	2,1	0	0,4	**95,9**	0,3	0,5	1956
Classe G	0,1	2,4	1,0	1,5	0,4	0,4	**91,9**	2,3	1071
Classe H	1,5	3,9	3,8	1,1	0,5	0,7	3,5	**85,1**	2544
Eficiência Média = 92,4									

Tabela 3.15: Matriz de confusão para o melhor classificador obtido pela metodologia AN3. Resultados para dados de *projeto* em porcentagem.

	Classe A	***Classe B***	***Classe C***	***Classe D***	***Classe E***	***Classe F***	***Classe G***	***Classe H***	***Total de Eventos***
Classe A	**96,7**	0	2,5	0,2	0,2	0,2	0	0	608
Classe B	0	**98,1**	0	0	0	0	0,6	1,3	858
Classe C	3,5	0,1	**95,3**	0	0	0,1	0,1	0,9	799
Classe D	0,1	0,7	0,1	**94,5**	0	0,1	2,8	1,7	768
Classe E	0	0	0,3	0,1	**98,9**	0	0,3	0,4	707
Classe F	0,3	0	0,3	0	0	**99,4**	0	0	733
Classe G	0	0,2	0,2	0,3	0	0	**98,6**	0,7	570
Classe H	0,5	0	0,2	0,7	0	0	1,7	**96,9**	848
Eficiência Média = 97,3									

empregando a metodologia AN2 e 97,3% e 96,7%, respectivamente, para os conjuntos de projeto e teste empregando a metodologia AN3.

Considerando-se os valores de eficiência contidos na Tabela 3.13, verifica-se que o pior resultado de classificação ocorre para espectros da classe H. Os espectros pertencentes a essa classe são confundidos, principalmente, com espectros pertencentes às classes D (1,7%) e G (1,2%). A única indicação desse resultado desfavorável vem dos resultados da distância mínima entre as curvas. Conforme pode ser verificado na Tabela 3.10, a curva representativa da classe D encontra-se a menor distância mínima da curva de H. O mesmo ocorre com relação às curvas das classes G e H. Os maiores porcentuais de confusão entre espectros, no entanto, são verifi-

Tabela 3.16: Matriz de confusão para o melhor classificador obtido pela metodologia AN3. Resultados para dados de *teste* em porcentagem.

	Classe A	***Classe B***	***Classe C***	***Classe D***	***Classe E***	***Classe F***	***Classe G***	***Classe H***	***Total de Eventos***
Classe A	**96,1**	0	3,1	0,2	0,2	0,3	0	0,2	608
Classe B	0	**98,1**	0	0	0	0	0,7	1,2	858
Classe C	3,7	0,1	**93,5**	0,3	0,1	0,3	0,1	1,9	1598
Classe D	0	0,4	0,4	**94,1**	0,1	0,3	3,1	1,6	768
Classe E	0,1	0	0,5	0	**98,4**	0	0,2	0,7	2828
Classe F	0,1	0	0,4	0,1	0	**99,2**	0,1	0	733
Classe G	0	0,4	0,6	0,2	0	0	**98,2**	0,6	500
Classe H	0,5	0,6	0,4	0,9	0	0,4	0,8	**96,5**	848
Eficiência Média = 96,7									

cados para as classes A e C. Conforme os resultados apresentados na Tabela 3.4, os centros dessas classes são bastante próximos, o que, provavelmente, provoca a confusão na classificação dos espectros das duas classes.

Analisando-se os resultados contidos na Tabela 3.14, verifica-se que, embora os resultados de classificação para dados de teste pertencentes à classe C sejam bons, os resultados para dados de teste estão dentre os piores (90,7%), indicando que a capacidade de generalização do classificador, para esta classe, não é boa. Supõe-se que um dos motivos da baixa capacidade de generalização observada com relação aos espectros dessa classe deve-se à divisão do conjunto de dados em conjuntos de projeto e de teste. Devido à complexidade envolvida na extração das curvas, decidiu-se, inicialmente, dividir os espectros de forma a obter conjuntos de teste com, aproximadamente, 1000 espectros cada. Como o conjunto de espectros da classe C é um dos maiores (veja a Tabela 3.1), o porcentual de espectros designados para o conjunto de projeto corresponde a, aproximadamente, 20% da quantidade total de espectros total enquanto que para a maior parte das classes esse porcentual é maior ou igual a 30%. Outro motivo que parece justificar a baixa capacidade de generalização dos espectros da classe C é a flutuabilidade dos dados verificada na análise estatística apresentada na Seção 3.2.1.

O emprego da metodologia AN3 permite que se obtenha um aumento significativo no desempenho do classificador. Dentre os resultados apresentados, verifica-se que a classe H é a que mais ganha em termos de eficiência de classificação quando se

passa de um classificador para o outro. Considerando os resultados para dados de teste verifica-se que o porcentual de acerto dessa classe passa de 85,1% para 96,5%. É possível que o emprego da média de dois espectros tenha sido suficiente para eliminar parte do ruído que fazia com que os espectros dessa classe se mesclassem com os espectros de outras.

Também para o classificador baseado na metodologia AN3, verifica-se gande porcentual de confusão entre espectros das classes A e C. Nota-se, também, que um porcentual significativo de espectros da classe B confunde-se com os espectros das classes G (classe cujo centro é o mais próximo do centro da classe B) e da classe H (classe de espectros mais próximos), resultado coerente com as Tabelas 3.4 e 3.5. Os resultados das análises das distâncias entre espectros e curvas e distâncias entre curvas, contidos nas Tabelas 3.9, 3.10 e 3.12 também são coerentes com os resultados de classificação citados. Observa-se, também, uma confusão significativa entre os espectros da classe D e os da classe G. Um indicativo desta possível confusão foi obtido através das medidas das distâncias de Hausdorff (Tabela 3.11).

A despeito do custo computacional do algoritmo de extração ser bastante significativo, a operação do classificador é razoavelmente simples, consistindo no cálculo da distância do espectro à curva representativa de cada classe, resultando num custo $O(kn)$ por curva, onde k e n representam o número de segmentos e a dimensão dos dados, respectivamente.

Os códigos-fonte das implementações do classificador em Matlab6 e em linguagem C são listados no Apêndice A. A versão listada foi compilada e executada num PC Athlon 2600 MHz/512KB, em ambiente Linux. O tempo gasto na classificação de um espectro, empregando as curvas principais e os conjuntos de teste foi igual a 2,8 ms. Apesar de operar com dados representados em 32 bits, enquanto que o classificador implementado em Matlab6 emprega dados representados com precisão de 64 bits, não houve redução na eficiência do classificador. Esses resultados são animadores e mostram que a tarefa de classificação de navios pode ser realizada num tempo muito curto com relação ao tamanho da janela de amostragem dos dados (186 ms), indicando a viabilidade de aplicação do método proposto na implementação de um sistema de reconhecimento de navios, baseado em sinais de sonar passivo, operando em tempo real.

3.5 Comparação do desempenho da classificação

Na Tabela 3.17, são comparados os resultados obtidos com outros trabalhos que utilizaram a mesma base de dados, adotando, no entanto, metodologias de projeto do classificador distintas, inclusive no que diz respeito à divisão dos dados nos conjuntos de projeto e de teste. Como os trabalhos citados empregaram apenas os espectros e não as médias de dois espectros, as eficiências médias informadas devem ser comparadas com os resultados obtidos para o método AN2, ao invés do melhor resultado, conseguido com o emprego do método AN3. Os resultados para redes neurais MLP e análise de componentes de discriminação (PCD) foram obtidos por Souza-Filho e Seixas [77]. Neste trabalho, os autores buscam, através do emprego de diversas técnicas de agrupamento (*clustering*), subconjuntos de dados que permitam o projeto de um classificador com boa eficiência e elevada capacidade de generalização. Após obtidos os subconjuntos procurados, é utilizado um classificador neural PCD seqüêncial [6], o qual emprega um número crescente de componentes de discriminação. Foi verificado pelos autores que, através do emprego de técnicas de agrupamento dos dados é possível obter um ganho considerável em termos de eficiência. De fato, os resultados apresentados são bastante expressivos, superando os obtidos com o classificador baseado em curvas principais. É possível que empregando os conjuntos de projeto e teste definidos por estes autores se obtenha uma melhora nos resultados obtidos empregando-se o classificador baseado em curvas principais. Por outro lado, empregando-se a metodologia AN3 (empregando-se como entrada do classificador não mais um espectro, mas a média de dois espectros sucessivos) se obteve uma eficiência média igual 96,7%, o que pode ser considerado como um resultado realmente expressivo. É possível que empregando-se esta abordagem a um classificador neural se obtenha resultados ainda melhores. Com o emprego de filtros casados, para 400 componentes principais, foi possível obter uma eficiência média de 90,5% [44]. Sendo esta uma técnica linear, ao contrário das redes neurais e curvas principais, era esperada uma eficiência média menor que a obtida pelos outros métodos.

Além da eficiência média de classificação, outras características também devem ser consideradas na escolha de um classificador. Devido à diversidade de missões realizadas por submarinos, em situações de patrulha ou em operações conjuntas,

Tabela 3.17: Comparação da eficiência de classificação de oito classes de navios empregando-se diversos métodos.

Método	**Eficiência (%)**
Curvas principais (método AN2)	92,4
Redes neurais MLP + PCD [77]	93,4 - 96,0
Filtros casados [44]	90,5

o sistema de classificação de navios empregado por estes meios deve operar com um número arbitrário de classes as quais, preferencialmente, devem constituir um banco de dados de assinaturas acústicas. Sendo constante a necessidade de inclusão e remoção de classes, é necessário que esses procedimentos possam ser realizados de forma simples e rápida. Outro requisito importante para este tipo de classificador é que ele possa se tornar mais especializado, inclusive em tempo de operação, através da inclusão de um ou mais níveis de decisão permitindo, por exemplo, que se identifique, além da classe, qual navio pertencente à classe é a fonte do ruído detectado. Embora o classificador baseado em curvas principais não tenha apresentado a melhor eficiência de classificação dentre os métodos empregados[8], ele ainda pode ser considerado como uma solução atraente para o problema de classificação específico, visto que atende aos requisitos particulares da aplicação, os quais, em geral, não são atendidos, na sua totalidade, por classificadores neurais não-especialistas [9]. Outra opção de classificador escalável e amplamente configurável é o baseado em filtros casados, proposto por Silva e Seixas [44]. No entanto, conforme discutido anteriormente, por se tratar de um classificador baseado em uma técnica linear, para o problema específico da classificação de sinais de sonar passivo, espera-se obter resultados de eficiência menores que os obtidos empregando-se técnicas não-lineares como, curvas principais, que podem extrair correlações de ordem superior, não acessíveis à técnica

[8]A eficiência média de classificação obtida para o método AN2 (empregando um espectro por vez) é menor que o limite inferior da faixa de valores de eficiência obtida para o classificador neural[77], no entanto, o resultado obtido para o método AN3, 96,7%, é melhor que o limite máximo obtido para o classificador neural.

[9]Deve-se considerar, no entanto, que, apesar da flexibilidade de configuração característica dos classificadores especialistas, em geral, após a inclusão de uma nova classe, é necessário o retreinamento de toda a rede especialista a fim de obter-se uma boa eficiência de classificação para todas as classes.

de componentes principais, que é a ferramenta básica dos filtros casados.

3.6 Sumário

Este trabalho propôs um classificador de navios utilizando sinais pré-processados de sonar passivo e a técnica de curvas principais. Através de um conjunto de curvas extraídas para cada classe, e utilizando um critério simples de decisão, foi possível obter eficiências superiores a 85% (classe H), totalizando, para o conjunto de teste, uma eficiência média de classificação igual a 92,4%. Empregando-se não mais os espectros singulares, mas a média de dois espectros sucessivos (método AN3), foi possível obter eficiências superiores a 93% (classe C), totalizando, para o conjunto de teste, uma eficiência média de classificação de 96,7%, resultado bastante significativo, em especial, pela complexidade do problema.

A utilização da técnica de curvas principais ao problema de sonar passivo resultou num classificador eficiente, escalável, facilmente configurável e de baixo custo computacional na fase de operação.

Tendo em vista os bons resultados encontrados, concluímos que esta técnica é capaz de resolver problemas complexos de representação e de classificação, tais como o focalizado neste trabalho.

Capítulo 4

Classificação de navios baseada em imagens infravermelhas

Neste capítulo, é proposto um classificador de navios que emprega curvas principais a fim de extrair informação relevante de imagens infravermelhas segmentadas dos mesmos. O conjunto de dados é composto por diversas vistas de cinco classes de navios, obtidas a partir de modelos tridimensionais, as quais simulam imagens obtidas por sensores infravermelhos empregados, por exemplo, por navios de superfície e submarinos. O módulo de classificação proposto é parte do esforço de desenvolvimento de equipamentos de vigilância baseados em sensores infravermelhos, os quais possuem crescente importância na defesa de diversos meios, tais como aeronaves, navios e tanques de combate. Como a tecnologia empregada na construção desses equipamentos é restrita a poucos países, seu custo é bastante elevado, o que limita seu emprego.

O sistema de reconhecimento de navios é composto por dois blocos principais: o módulo de detecção e segmentação de imagens e o módulo de extração de características e classificação. Este capítulo trata do segundo módulo, o qual será discutido em detalhe.

A construção do classificador prevê a extração da curva principal representativa de cada classe de navio que se deseja identificar. Essas curvas compõem o banco de dados de características de navios. Quando uma imagem segmentada, proveniente do módulo de detecção e segmentação de imagens, é recebida, suas características discriminantes são extraídas. Serão abordados vários métodos de obtenção de carac-

terísticas baseados nas informações de contorno externo das vistas. É conveniente supor que essas características, extraídas sob a forma de cotas, constituem-se nas coordenadas de um ponto num espaço multidimensional, onde são definidas as curvas principais. Adotando-se como critério de decisão atribuir a imagem segmentada recebida pelo sistema classificador à classe cuja curva representativa está a menor distância euclidiana do ponto representado pelas suas características, foi possível obter uma eficiência média de classificação igual a 97,3%, resultado significativamente melhor que o obtido anteriormente para um classificador neural que empregou, como características discriminantes, momentos invariantes extraídos das imagens [78].

Com o intuito de analisar a robustez do classificador, foram gerados conjuntos de teste ruidosos, os quais foram apresentados ao classificador. Os resultados de classificação indicam que o classificador é razoavelmente robusto ao tipo de ruído inserido.

4.1 Introdução

Uma imagem infravermelha, conforme exemplificado na Figura 4.1, a qual mostra a imagem de um navio de superfície captada por um detector específico, é um mapeamento bidimensional, em tons de cinza, que representa a parcela da energia infravermelha irradiada por um corpo e o ambiente ao seu redor [79]. Conforme pode ser observado, empregando-se a tecnologia atual, não é possível obter imagens tão detalhadas quanto as geradas por sensores óticos. Além disso, esse tipo de imagem, usualmente, não possui muito contraste. Ainda assim, as informações extraídas empregando-se sensores infravermelhos possuem aplicações na indústria, em particular, na indústria bélica e em diversas áreas, como a astronomia e a monitoração por satélite do meio ambiente.

A detecção antecipada e a classificação de ameaças são de importância crítica para a defesa dos meios navais. A fim de realizar tais tarefas, esses meios dispõem de vários equipamentos dotados de diferentes sensores. Dois tipos de armamento são especialmente considerados de difícil detecção e alta periculosidade: os mísseis guiados pelo calor irradiado pelo alvo e, no caso específico dos navios de superfície, os mísseis que se deslocam muito próximo à superfície do mar (*sea skimming missi-*

Figura 4.1: Imagem infravermelha de um navio de superfície.

les). Tanto um como outro não podem ser facilmente detectados por equipamentos convencionalmente empregados nas tarefas de vigilância, tais como radares e equipamentos de MAGE (Medidas de Apoio a Guerra Eletrônica), os quais, normalmente, operam na faixa de freqüência que abrange as bandas C a K (0,5GHz a 40GHz) [80]. Além de câmeras que operam na faixa do espectro visível, equipamentos FLIR (*Forward-Looking InfraRed*) [1] e de MAGE-IV (MAGE infravermelho) [2] foram introduzidos no mercado a fim de suprir a demanda por sistemas de defesa capazes de proteger um navio ou aeronave destas ameaças. Estes equipamentos detectam irradiações na faixa do infravermelho, usualmente operando na faixa do infravermelho médio (MIR - *middle infrared*), o qual compreende os comprimentos de onda na região de 3μm (100THz) a 6μm(50THz), ou na região do infravermelho distante, faixa que compreende comprimentos de onda de 6μm(50THz) a 15μm(20THz). Para

[1] O FLIR é um equipamento de vigilância, comumente empregado em aeronaves, dotado de sensores infravermelhos, cujo objetivo é o de fornecer um alarme, em tempo hábil, da presença de uma ameaça (como, por exemplo, a aproximação de um míssil guiado por calor) à plataforma onde o equipamento se encontra instalado.

[2] Várias informações sobre equipamentos de FLIR, MAGE-IV e sensores infravermelhos podem ser obtidas nas páginas eletrônicas dos fabricantes. Veja, por exemplo, as páginas das empresas FLIR Inc., *http://www.flir.com/imaging*; HGH Systemes Infrarouges, *http://www.hgh.fr/survaillance-defense-en.php*; e Thales Optronic Systems, *http://www.thalesgroup-optronics.com/dos/default.shtml*.

serem eficazes, esses equipamentos precisam possuir um tempo de resposta muito pequeno (usualmente, menor que 1 segundo), sendo, portanto, necessário que as tarefas de detecção e classificação sejam realizadas de forma automática, gerando uma alarme que permitirá aos membros da tripulação tomar medidas que visam eliminar ou neutralizar a ameaça.

Submarinos navegando na superfície ou na cota periscópica (próximo a superfície, a uma profundidade que permite a observação através do periscópio) dependem de câmeras de diversos tipos a fim de monitorar o ambiente ao seu redor. Durante a noite ou em situações de baixa visibilidade, é desejável o emprego de equipamentos baseados em sensores infravermelhos, a fim de manter a vigilância, independente das condições de visibilidade. No entanto, mesmo operadores treinados possuem dificuldade em classificar navios a partir de imagens infravermelhas, visto que este tipo de imagem não possui boa definição e contraste, impedindo que se percebam detalhes do navio que está sendo observado. Por esta razão, nesses casos, é desejável que o operador disponha de um classificador automático como ferramenta de apoio à decisão.

Alguns trabalhos encontrados na literatura abordam a detecção e a classificação de alvos baseada em imagens infravermelhas. Em [81] (1996) Sadjadi e Chun empregam imagens sintéticas do míssel russo *scud* e seu lançador, além de imagens reais de um cubo, obtidas a partir de um sensor infravermelho com diferentes tipos de polarização linear. Tanto os sinais reais como os sintéticos são compostos por três matrizes que representam o porcentual de polarização, o ângulo de polarização e a de intensidade dos pixels da imagem infravermelha. As informações de polarização são obtidas, algebricamente, através dos parâmetros de Stroke do sinal captado pelos polarizadores. As bordas do alvo são extraídas empregando-se técnicas convencionais sobre as imagens infravermelhas, enquanto que as regiões ocupadas pelo objeto são extraídas a partir das informações de polarização, empregando-se uma técnica baseada em análise de textura, proposta pelos autores. Em seguida, são aplicadas técnicas estatísticas de reconhecimento de padrões. Os autores concluem que adicionando informações de polarização às características extraídas das imagens infravermelhas é possível aumentar significativamente a eficiência do sistema de classificação. Em [82] (2001) são fornecidos mais detalhes sobre o estudo iniciado pelos

autores em [81].

Em [83] (1996) Haddon, Boyce e Strens apresentam uma técnica de classificação de diferentes tipos de cobertura de terrenos a partir de seqüências de imagens infravermelhas obtidas por uma aeronave. A textura das imagens é analisada empregando-se polinômios de Hermite, cujos coeficientes mais discriminantes servem como entrada para uma rede neural, a qual realiza uma primeira fase de classificação. A fase final de classificação é realizada por um algoritmo baseado em teoria da informação, tendo os autores obtido uma eficiência de classificação de cinco tipos distintos de cobertura de terreno igual a 90%.

Kotrlick e Real [84] (2000) propõem um sistema de alarme antecipado baseado em sensores infravermelhos aplicados à defesa de carros de combate. São apresentadas as técnicas de processamento dos sinais durante as fases de detecção, extração de características relavantes e classificação dos alvos, sendo esta última fase realizada empregando-se o discriminante linear de Fisher [3]. Os autores empregam dados reais, divididos em três classes distintas, tendo obtido uma eficiência de classificação igual a 100%.

Outro trabalho abordando tema correlato foi desenvolvido por Amaral [78] (2001). Inicialmente trabalhando com dados sintéticos e, em seguida, com algumas vistas reais, o autor emprega uma rede neural MLP, treinada pelo algoritmo *back-propagation*, a fim de classificar cinco classes de navios. Como entradas para a rede neural são empregados momentos invariantes [85] extraídos das imagens, obtendo-se, para dados sintéticos, uma eficiência média de classificação igual a 87,3%.

Bharadwaj e Carin [86] (2002) propõem um classificador baseado em modelos de Markov. O conjunto de dados empregado consiste de imagens infravermelhas de quatro classes de carros de combate, obtidas por sistemas FLIR. A imagem é dividida em regiões, as quais são comparadas com modelos obtidos a partir dos dados de projeto. Os autores verificam que o método proposto produz melhores resultados que abordagens alternativas baseadas na transformada de Radon [85] e em quantização vetorial [27]. A eficiência média de classificação, considerando a abordagem proposta, é de 90%.

O trabalho apresentado por Neves, da Silva e Mendonça [87] (2003), trata da segmentação de imagens infravermelhas empregando uma técnica que combina

morfologia matemática e a transformada wavelet de módulo máximo, a fim de obter um sistema automático de detecção e segmentação de vistas de navios, o qual pode ser empregado, por exemplo, no pré-processamento de imagens infravermelhas para um sistema de classificação e alarme antecipado. O sistema proposto possui a habilidade de extrair contornos fechados de objetos mais quentes ou mais frios que o ambiente, opera bem em situações de baixo contraste entre o objeto e o meio e produz imagens desprovidas da linha d'agua.

O classificador proposto neste capítulo utiliza um sistema dotado de sensores infravermelhos para identificar cinco classes distintas de navios de superfície. Das imagens segmentadas, apenas se extrairá informações dos contornos, não se considerando outros aspectos. Assim, supõe-se que essa informação é suficiente para que se obtenha uma boa discriminação das classes de navios. Conforme será exposto na Seção 4.2, os modelos tridimensionais de navios gerados por Amaral [78] foram empregados na geração dos conjuntos de dados usados no desenvolvimento do classificador de navios baseado em curvas principais.

Muito embora o classificador proposto tenha sido desenvolvido visando sua aplicação em um sistema baseado em sensores infravermelhos, os métodos de extração de informação a serem discutidos nas seções subsequentes podem ser aplicados, sem qualquer modificação, a imagens segmentadas, obtidas por sensores óticos. Uma aplicação possível para um classificador deste tipo, no âmbito naval, poderia dar-se como um sistema de apoio na identificação de alvos por um submarino. Usualmente, este tipo de navio mantém-se submerso, vindo a cota periscópica apenas por um breve período de tempo, durante o qual seu periscópio é içado para que se monitore o ambiente ao seu redor. Um sistema que capturasse as imagens obtidas, segmentasse os objetos de interesse e os classificasse poderia tornar a tarefa de vigilância mais rápida e segura.

4.2 O conjunto de dados

Cinco classes distintas de navios foram modeladas em Matlab5.3.0 por Amaral [78], empregando uma série de planos (*multifaceted patches*) definidos a partir de matrizes de vértices e de faces. A matriz de vértices possui as coordenadas dos

vértices dos planos que compõem o modelo, enquanto que, cada linha da matriz de faces indica a seqüência de vértices que define cada plano do modelo.

As classes de navios empregadas – porta-aviões, contratorpedeiro, fragata, navio-tanque e navio-patrulha – foram modeladas, de forma simplificada, embora tendo sido consideradas as dimensões originais de navios reais.

Para cada classe de navio, dorovante designadas pelas letras A (porta-aviões), B (contratorpedeiro), C (fragata), D (navio-tanque) e E (navio-patrulha), foi empregado um arquivo contendo as respectivas matrizes de vértices e de faces. A partir desses arquivos, foram produzidos, para cada classe de navio, um conjunto de 8280 vistas distintas. As vistas foram geradas empregando-se os recursos do Toolbox de Visualização 3D do Matlab6(R12), considerando a distância entre o navio e o observador igual a 2.500 jardas, ângulos de azimute na faixa de -90^o a 89^o e ângulos de elevação variando de 0^o a 45^o, ambos definidos em intervalos de 1^o.

A divisão dos dados em conjuntos de projeto e de teste seguiu o mesmo critério adotado por Amaral [78]. Os conjuntos de projeto foram empregados no desenvolvimento do classificador, incluindo a extração das curvas principais, enquanto que o conjunto de teste foi utilizado na verificação do desempenho e da capacidade de generalização do classificador. O critério de divisão dos dados tornou possível comparar, diretamente, os resultados obtidos com os apresentados em [78]. Os conjuntos de projeto, os quais são empregados na extração das curvas principais das classes, consistem de 144 vistas de cada classe de navio, compreendendo as vistas com ângulos de azimute nas faixas de -90^o a -45^o, em intervalos de 5^o; -30^o a 30^o, em intervalos de 15^o; e 45^o a 85^o, em intervalos de 5^o e ângulos de elevação iguais a 0^o, 7^o, 15^o, 22^o, 30^o e 45^o. Os conjuntos de teste das classes compreenderam todas as 8280 vistas.

A seguir são apresentadas diversas figuras contendo uma foto de um navio da classe modelada e algumas vistas de cada uma das classes. As Figuras 4.2, 4.4, 4.6, 4.8 e 4.10 mostram, respectivamente, um porta-aviões, um contratorpedeiro, uma fragata, um navio patrulha e um navio-tanque. As Figuras 4.3, 4.5, 4.7, 4.9 e 4.11 exibem quatro vistas de cada uma das classes, para diferentes ângulos de azimute e elevação. É conveniente observar que estas vistas foram geradas considerando-se distâncias entre o observador e o navio, ajustadas caso a caso, a fim de permitir uma

melhor visualização dos detalhes.

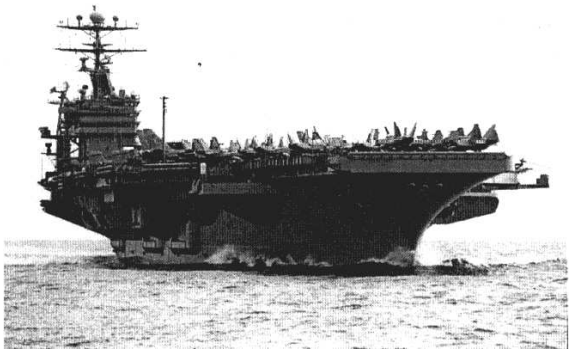
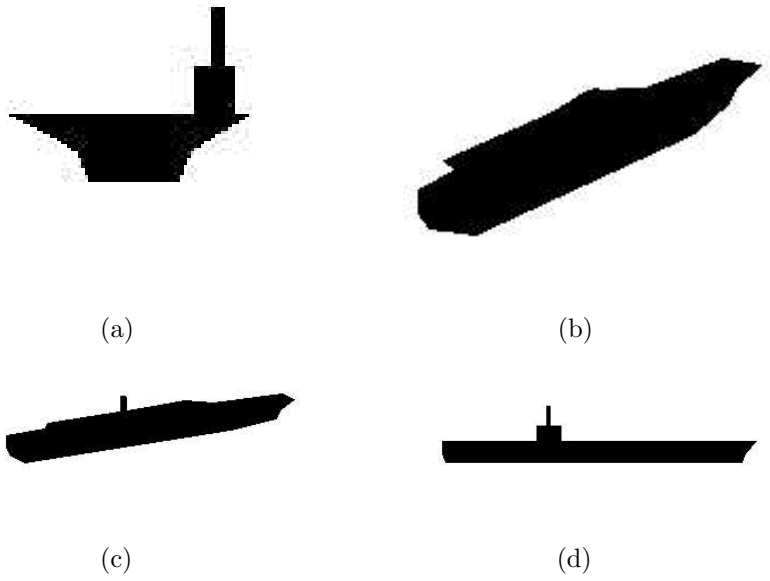


Figura 4.2: Foto de um porta-aviões de forma e dimensões compatíveis com o modelo da classe.



(a) (b)

(c) (d)

Figura 4.3: Vistas da classe A (porta-aviões) para os seguintes ângulos de azimute (az) e elevação (el): (a) az $= -90^o$, el $= 0^o$; (b) az $= -60^o$, el $= 15^o$; az $= -30^o$, el $= 15^o$ e az $= 0^o$, el $= 0^o$.

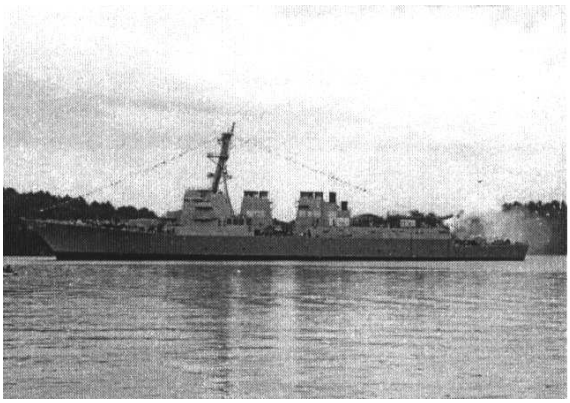


Figura 4.4: Foto de um contratorpedeiro de forma e dimensões compatíveis com o modelo da classe.

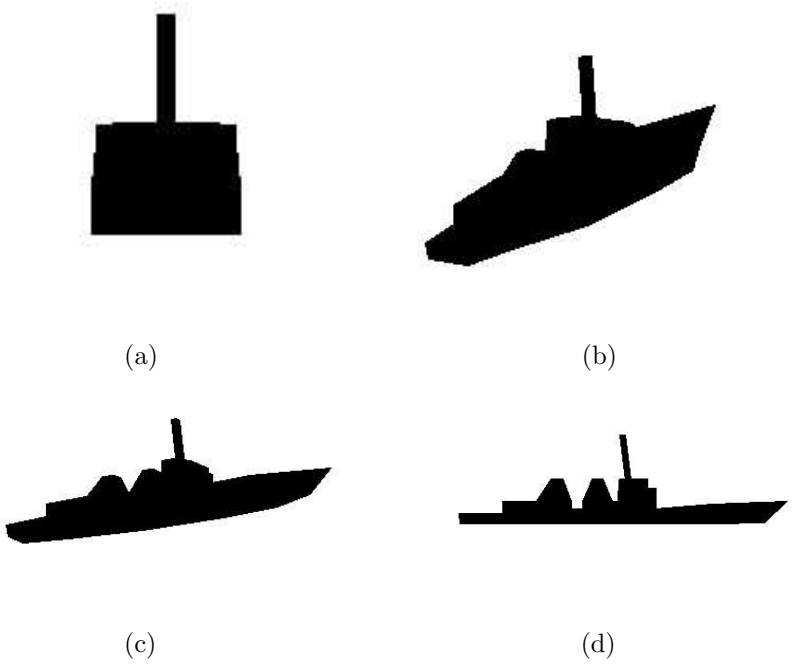
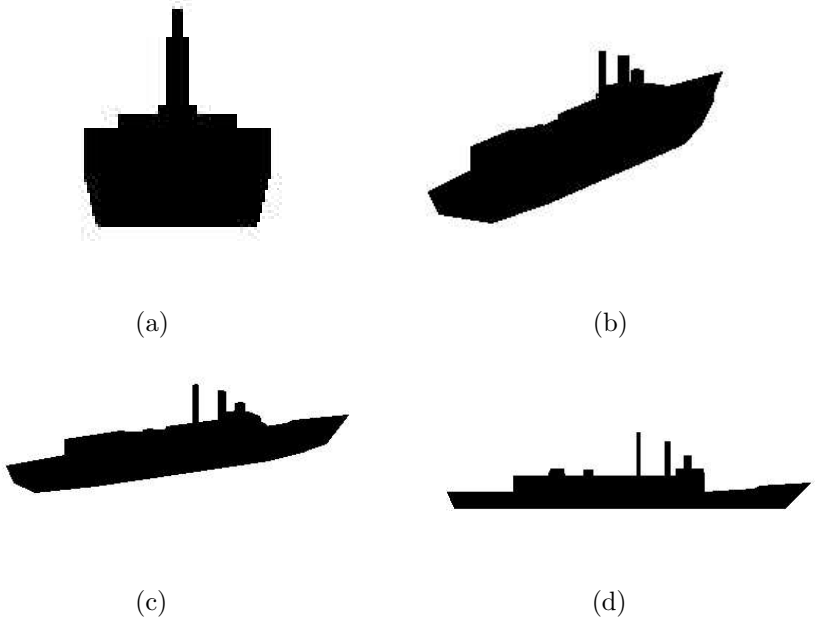


Figura 4.5: Vistas da classe B (contratorpedeiro) para os seguintes ângulos de azimute (az) e elevação (el): (a) az = -90^o, el = 0^o; (b) az = -60^o, el = 15^o; az = -30^o, el = 15^o e az = 0^o, el = 0^o.

Figura 4.6: Foto de uma fragata de forma e dimensões compatíveis com o modelo da classe.

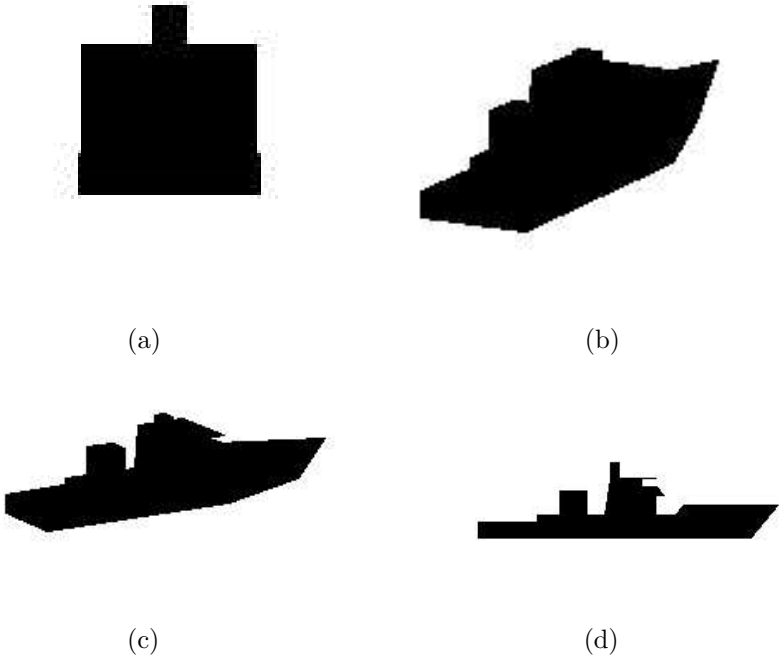


(a) (b)

(c) (d)

Figura 4.7: Vistas da classe C (fragata) para os seguintes ângulos de azimute (az) e elevação (el): (a) az $= -90^o$, el $= 0^o$; (b) az $= -60^o$, el $= 15^o$; az $= -30^o$, el $= 15^o$ e az $= 0^o$, el $= 0^o$.

Figura 4.8: Foto de um navio-patrulha de forma e dimensões compatíveis com o modelo da classe.

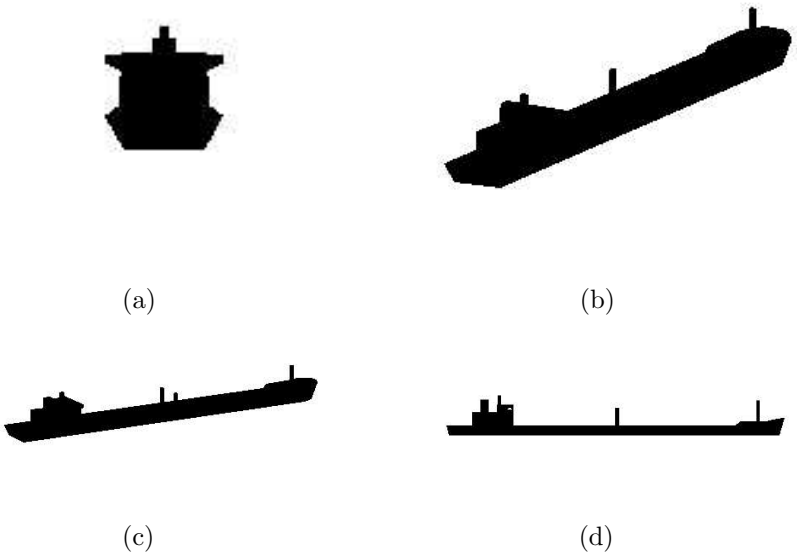


(a) (b)

(c) (d)

Figura 4.9: Vistas da classe D (navio-patrulha) para os seguintes ângulos de azimute (az) e elevação (el): (a) az $= -90^o$, el $= 0^o$; (b) az $= -60^o$, el $= 15^o$; az $= -30^o$, el $= 15^o$ e az $= 0^o$, el $= 0^o$.

Figura 4.10: Foto de um navio-tanque de forma e dimensões compatíveis com o modelo da classe.



(a) (b)

(c) (d)

Figura 4.11: Vistas da classe E (navio-tanque) para os seguintes ângulos de azimute (az) e elevação (el): (a) az $= -90^o$, el $= 0^o$; (b) az $= -60^o$, el $= 15^o$; az $= -30^o$, el $= 15^o$ e az $= 0^o$, el $= 0^o$.

A Tabela 4.1 contém a quantidade de vértices e planos empregados na construção dos modelo de cada classe de navio [78]. Comparando-se a quantidade de planos empregada na construção de cada modelo, pode-se observar que o modelo da classe C é o mais detalhado, enquanto que o da classe E é o que emprega o menor número de planos.

Tabela 4.1: Quantidade de vértices e planos empregados na construção dos modelos das cinco classes de navios.

	Classe A	**Classe B**	**Classe C**	**Classe D**	**Classe E**
Número de vértices	45	92	130	92	76
Número de planos	34	57	66	57	32

4.3 Extração de características discriminantes

Conforme exposto na Seção 4.2, apenas os contornos externos das vistas foram considerados como características discriminantes dos navios das diferentes classes. A detecção da vista no quadro da imagem é realizada verificando-se a amplitude associada aos pixels, visto que o fundo é branco e a vista é preta. A extração dos pontos de contorno, aonde aplicável, foi realizada empregando-se o algoritmo de trilhamento de contorno, descrito na Figura 9.15 de [85].

Cinco metodologias distintas de mapeamento de contorno foram testadas. Todos os métodos empregam um número fixo de cotas, onde cada valor de cota representa a distância, em pixels, de um ponto do contorno da vista a uma das bordas do quadro ou a um ponto específico. Todas as cotas são normalizadas, possuindo valores no intervalo $[0, 1]$. Inicialmente, as cotas são extraídas; em seguida, subtrai-se, de cada cota, o menor valor de cota e, por fim, dividem-se todos os valores obtidos pelo maior valor resultante da operação anterior. Em termos práticos, considerando-se que uma vista representa o resultado da segmentação da imagem de um navio, a normalização garante que os valores de cotas são independentes da distância entre o sensor e o navio observado.

A quantidade de cotas é igual a 300 ou 400, de acordo com o método empregado, e esta quantidade foi definida a fim de obter-se um mapeamento detalhado das vistas. Conforme será abordado em detalhe nos parágrafos subsequentes, os métodos

referidos como I, II e IV são empregados no mapeamento do contorno superior ou de uma das laterais das vistas. Como cada vista é representada por uma matriz de pixels de dimensão 420 x 560 (linhas x colunas), empregando-se 300 cotas verificou-se que, para a maioria das vistas geradas, todos os pontos do contorno superior ou lateral das vistas seriam mapeados, maximizando-se, desta forma, os resultados de eficiência de classificação. Os métodos III e V são empregados no mapeamento de extensões maiores, e, por esta razão, empregam 400 cotas. Idealmente, todos os pontos do contorno definido deveriam ser considerados; no entanto, quanto maior o número de cotas, maior a carga computacional envolvida tanto na fase de extração de cotas como na de classificação. Além disso, quanto maior o número de cotas, maiores serão os arquivos que contêm as matrizes de coordenadas dos vértices (*vertices*) e de interligação entre vértices (*edges*) das curvas principais que representam as classes envolvidas.

O intervalo entre cotas adjacentes deve ser um número inteiro, obtido em função da quantidade de cotas e do número de pontos que serão mapeados. Juntamente com a descrição de cada método de mapeamento, será apresentada a forma de obtenção do intervalo entre cotas. Entretanto, algumas questões comuns serão abordadas nos dois parágrafos seguintes.

Embora o incremento (em pixels ou em pontos, dependendo do método) entre cotas adjacentes deva ser inteiro, genericamente, este valor é fracionário, visto que é obtido pela razão entre o número de cotas a serem extraídas e a quantidade de pontos a serem mapeados. Assim, cada vez que se toma uma cota, de índice arbitrário i, o algoritmo de mapeamento calcula, em função do intervalo genérico entre cotas, o deslocamento $d(i)$ (também medido em pixels ou em pontos) em relação ao ponto de onde se iniciou o mapeamento. O valor de deslocamento, após arredondado para o inteiro mais próximo, torna-se a distância entre o primeiro e o i-ésimo ponto de mapeamento $D(i)$. Se esse valor é igual ao valor de distância obtida para a cota anterior, ou seja, se $D(i) = D(i-1)$, assume-se que o incremento, neste caso, é igual a zero e, assim, não há deslocamento e o valor da i-ésima cota é igual ao da cota anterior. A situação descrita ocorre quando o número de cotas ultrapassa a quantidade de pontos que se deve mapear, sendo necessário que um ou mais pontos do contorno sejam empregados duas ou mais vezes durante o processo

de mapeamento. Por outro lado, se a quantidade de pontos a serem mapeados é maior que a quantidade de cotas, perde-se detalhes da vista, o que pode implicar numa redução na eficiência de classificação. Empregando-se um valor de incremento fracionário conforme descrito, faz-se com que o mapeamento seja, aproximadamente, uniforme ao longo da área a ser mapeada.

A Figura 4.12 mostra um quadro de uma imagem que contém uma vista arbitrária. Os pixels pertencentes ao quadro são representados por meio de um sistema de coordenadas x e y cuja origem O é o ponto de coordenadas $x_O = 1$ e $y_O = 1$, ou seja, $(1, 1)$. Os pontos mostrados são frequentemente referenciados nos parágrafos subsequentes, os quais tratam dos métodos de mapeamento empregados na extração de características discriminantes das vistas de navios. Os seguintes pontos pertencem às vistas:

A_1**:** ponto de maior valor de coordenada y (ponto mais abaixo);

A_2**:** ponto de menor valor de coordenada x (ponto mais a esquerda);

A_3**:** ponto de maior valor de coordenada x (ponto mais a direita).

Os pontos abaixo descritos pertencem à primeira linha ou à primeira coluna do quadro e correspondem aos pontos A_1, A_2 e A_3:

P_{1y}**:** ponto da borda lateral esquerda (primeira coluna da matriz de pixels), situado a menor distância do ponto A_1;

P_{2y}**:** ponto da borda lateral esquerda, situado a menor distância do ponto A_2;

P_{2x}**:** ponto da borda superior (primeira linha da matriz de pixels), situado a menor distância do ponto A_2;

P_{3x}**:** ponto da borda superior, situado a menor distância do ponto A_3.

4.3.1 Mapeamento do tipo I

O método I emprega 300 cotas verticais, igualmente espaçadas, que mapeiam o contorno superior das vistas. As distâncias entre cotas são tomadas, em pixels, partindo-se de pontos da borda superior do quadro a pontos do contorno superior da

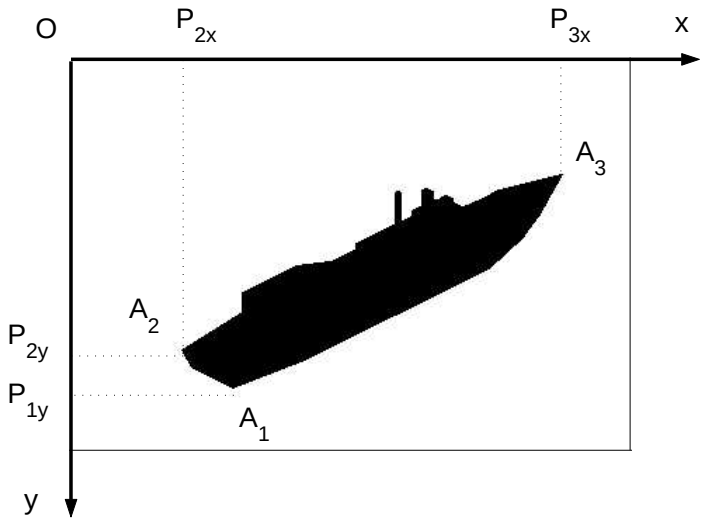
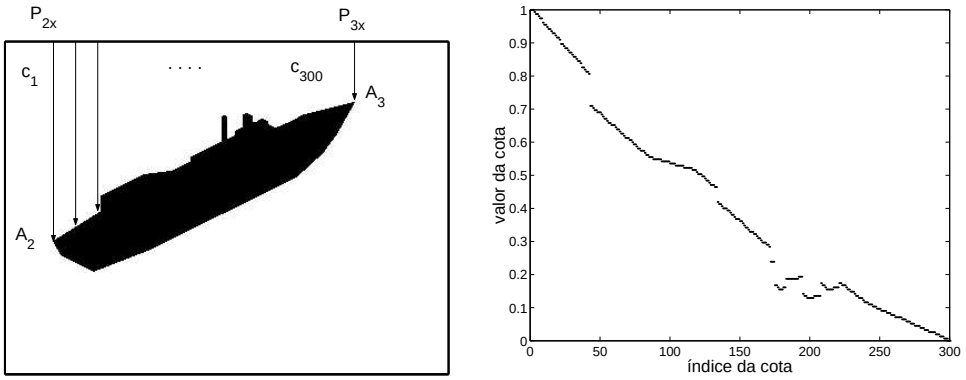


Figura 4.12: Quadro contendo uma vista arbitrária e pontos de referência.

vista, conforme esquematizado na Figura 4.13(a). A primeira cota c_1 representa a distância do ponto P_{2x} ao ponto A_2 (cujo valor corresponde ao valor da coordenada y do ponto A_2) enquanto que o valor da última cota c_{300} é igual à distância do ponto P_{3x} ao ponto A_3 (a qual corresponde ao valor da coordenada y do ponto A_3). A Figura 4.13(b) exibe um gráfico contendo o valor das cotas extraídas da vista mostrada. O eixo das abcissas representa o índice das cotas, cujos valores estão contidos no intervalo $[1, 300]$, enquanto que o eixo das ordenadas representa o valor de cada cota, o qual corresponde ao valor da coordenada y do ponto mapeado pertencente ao contorno superior da vista. Note que as cotas são normalizadas, possuindo valores contidos no intervalo $[0, 1]$. Conforme pode ser observado na Figura 4.13(a), a primeira cota, c_1, cujo índice é igual a 1, é a maior, por isso, seu valor, conforme visto na Figura 4.13(b) é unitário. A cota de índice 300, c_{300}, neste caso, é a menor e, por este motivo, possui valor nulo. As demais cotas possuem comprimentos intermediários e, por isso, assumem valores contidos no intervalo $(0, 1)$.

O intervalo entre cotas no mapeamento do tipo I, Δ_I, é obtido como $\Delta_I = n/300$, onde n é o número de pixels, contados sobre a borda superior do quadro, compreendidos entre os pontos P_{2x} e P_{3x}. A fim de calcular a distância do ponto P_{2x} ao ponto de onde se deve obter a i-ésima cota, $D(i)$, aplica-se a seguinte equação: $D(i) = round(i * \Delta_I)$, onde a função $round$ obtém o valor inteiro mais próximo do



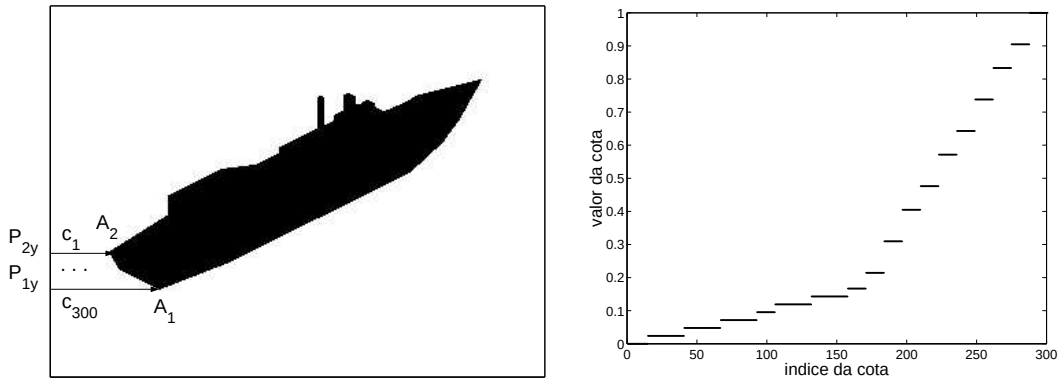
(a) (b)

Figura 4.13: Mapeamento do tipo I: (a) esquema de extração de cotas (b) cotas extraídas para a vista exibida em (a).

seu argumento e D não pode assumir um valor maior que o equivalente à distância entre os pontos P_{2x} e P_{3x}. Após a extração, as cotas são normalizadas e irão compor o conjunto de características discriminantes extraídas pelo método de mapeamento I.

4.3.2 Mapeamento do tipo II

O mapeamento do tipo II emprega 300 cotas horizontais a fim de mapear um dos lados da vista. Inicialmente, o algoritmo encontra os pontos A_1, A_2, P_{1y} e P_{2y}, conforme mostrado na Figura 4.14. O mapeamento se dá do ponto P_{2y} ao ponto P_{1y}, sendo o valor da primeira cota, c_1, igual à distância, em pixels, entre o ponto P_{2y} e o ponto A_2, enquanto o valor da última cota, c_{300}, é igual a distância, em pixels, entre os pontos P_{1y} e A_1. A distância entre cotas adjacentes é mantida, aproximadamente, uniforme e é calculada da mesma forma que no caso do método I, apenas considerando os pontos P_{2y} e P_{1y} ao invés de P_{2x} e P_{3x}, ou seja, a distância do ponto P_{2y} ao ponto de onde se obtém a i-ésima cota, $D(i)$, é obtido como $D(i) = round(i * \Delta_{II})$, onde $\Delta_{II} = n/300$ e n é a distância, em pixels, entre os pontos P_{2y} e P_{1y}, tomada sobre a borda lateral que contém os pontos. Após a extração, as cotas são normalizadas, ficando seus valores compreendidos no intervalo [0,1]. As cotas extraídas dessa forma compõem o conjunto de características discriminantes obtidas pelo o método de mapeamento II.

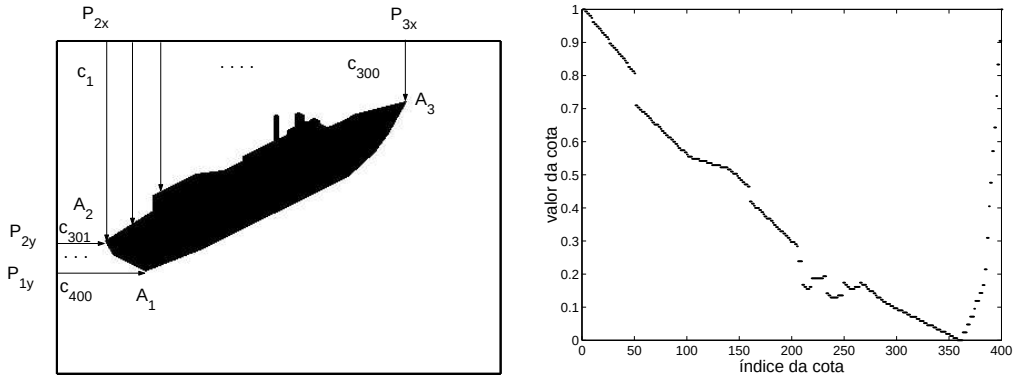


(a) (b)

Figura 4.14: Mapeamento do tipo II: (a) esquema de extração de cotas (b) cotas extraídas para a vista exibida em (a).

4.3.3 Mapeamento do tipo III

O mapeamento do tipo III é uma mistura dos mapeamentos dos tipos I e II. Conforme mostrado na Figura 4.15, são obtidas 300 cotas da parte superior da vista, partindo do ponto P_{2x} até o ponto P_{3x}, o que corresponde ao mapeamento do contorno da vista desde o ponto A_2 até o ponto A_3 e, em seguida, são obtidas 100 cotas da parte lateral da vista, indo desde o ponto P_{2y} ao ponto P_{1y}, mapeando o contorno entre os pontos A_1 e A_2. Após normalizadas, as cotas tomadas de todas as vistas de uma classe formam o conjunto de características extraídas conforme o método de mapeamento III.



(a) (b)

Figura 4.15: Mapeamento do tipo III: (a) esquema de extração de cotas (b) cotas extraídas para a vista exibida em (a).

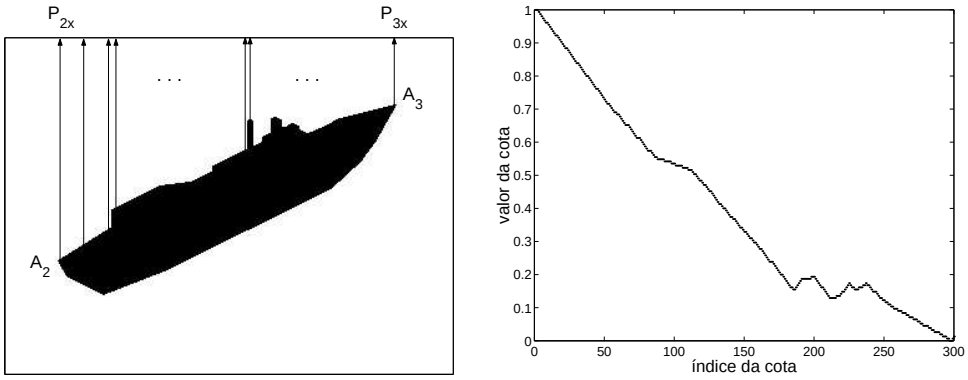
4.3.4 Mapeamento do tipo IV

Este tipo de mapeamento visa obter 300 cotas que partem do contorno superior da vista, do ponto A_2 ao ponto A_3, até a borda superior do quadro, conforme pode ser visto na Figura 4.16. Inicialmente, é obtido o valor da coordenada y do ponto A_2 (a qual representa a distância, em pixels, deste ponto à borda superior do quadro) e, em seguida, são obtidos, de forma ordenada, os valores das coordenadas y dos demais pontos do contorno superior da vista compreendidos entre este ponto e o ponto A_3. Esses n valores são armazenados em um array de onde serão extraídas as cotas procuradas. A primeira cota, c_1, é igual ao primeiro valor armazenado no array, o qual corresponde à coordenada y do ponto A_2. O índice do array correspondente à i-ésima cota $ind(i)$ é obtido como $ind(i) = round(i * \Delta_{IV})$ onde $\Delta_{IV} = n/300$, n é a quantidade de pontos compreendidos entre os pontos A_2 e A_3 e $ind \leq 300$, $\forall\, i$.

É importante observar que, embora os métodos I e IV mapeiem a parte superior das vistas, no primeiro caso, as cotas são extraídas a partir dos pixels do topo do quadro, a intervalos definidos pela quantidade de pixels contidos no intervalo entre os pontos P_{2x} e P_{3x} enquanto que, no segundo caso, o mapeamento é realizado a partir do contorno da vista, seguindo os pontos de contorno extraídos e, tomando cotas, a intervalos definidos em função da quantidade de pixels que compõem esse trecho do contorno, definido desde o ponto A_2 ao ponto A_3. Conforme pode ser observado, comparando-se as Figuras 4.13(b) e 4.16(b), empregando-se o mapeamento do tipo IV obtém-se cotas mais uniformes, sendo possível explorar melhor os detalhes do contorno. Intuitivamente, supôs-se que o método de mapeamento IV resultaria em curvas mais representativas para cada classe, o que acarretaria numa maior eficiência do classificador. No entanto, conforme será apresentado na Seção 4.5, os resultados de eficiência dos classificadores baseados nos dois métodos mostraram-se equivalentes.

4.3.5 Mapeamento do tipo V

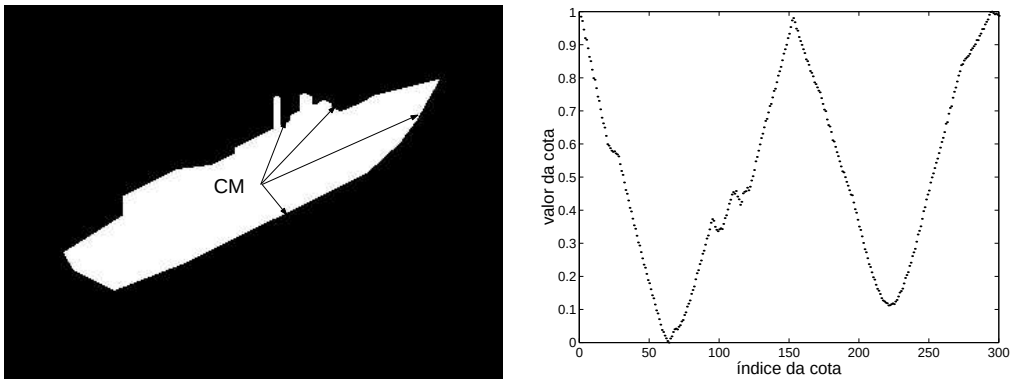
O método V emprega 400 quotas a fim de mapear todo o contorno da vista, conforme ilustrado na Figura 4.17(a). Inicialmente, são obtidos e armazenados em um array todos os pontos do contorno externo da vista e, a partir dos valores das



(a) (b)

Figura 4.16: Mapeamento do tipo IV: (a) esquema de extração de cotas (b) cotas extraídas para a vista exibida em (a).

coordenadas destes pontos, são calculadas as coordenadas do centro de massa (CM) da vista [85]. As cotas são tomadas do centro de massa aos pontos de contorno da vista, iniciando-se pelo ponto A_2. O índice do array correspondente à i-ésima cota $ind(i)$ é obtido de como $ind(i) = round(i * \Delta_V)$ onde $\Delta_V = n/400$, n é a quantidade total de pontos de contorno da vista e $ind \leq 400$, $\forall\, i$. As cotas extraídas são então normalizadas, conforme pode ser observado na Figura 4.17(b), o que faz com que os valores obtidos não dependam do tamanho da vista, mas apenas da forma do contorno, o que, na prática, torna os valores das cota independentes da distância entre o navio e a câmera.



(a) (b)

Figura 4.17: Mapeamento do tipo V: (a) esquema de extração de cotas (b) cotas extraídas para a vista exibida em (a).

4.4 Implementação

A Figura 4.18 apresenta um diagrama de blocos esquemático de um sistema que contém o classificador de navios. O primeiro bloco representa um sistema de detecção de um navio. Sua função é a de manter a vigilância do ambiente ao redor da plataforma, detectar a presença de um navio nas proximidades e enviar uma ou mais imagens do navio para o bloco seguinte. Estas tarefas podem ser desempenhadas, manualmente, por um operador ou por um sistema automático que percebe a mudança no cenário. A seguir, o módulo de segmentação detecta o navio presente na imagem e realiza o processo de segmentação da vista. A imagem segmentada é então fornecida ao módulo de classificação, o qual extrai as cotas necessárias e classifica o navio com base nas curvas principais armazenadas em seu banco de dados.

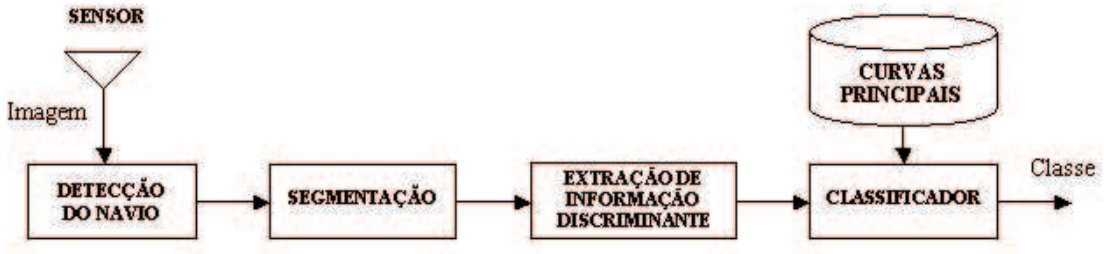


Figura 4.18: Diagrama de blocos do sistema de detecção e classificação de vistas de navios baseado em curvas principais.

O classificador empregado para o caso das vistas de navios é similar ao apresentado na Seção 3.3.1 para o caso da classificação de sinais de sonar passivo, cuja abordagem é baseada em [34]. Neste trabalho são implementados os módulos de extração de informação discriminante e o classificador. Ao invés de imagens segmentadas, foram empregadas as vistas sintéticas abordadas na Seção 4.2. Conforme discutido na Seção 4.3, cinco formas distintas de extração de informação discriminante foram implementadas, tendo sido gerados cinco de curvas principais para cada classe.

4.4.1 O classificador de vistas de navios baseado em curvas principais

Cinco classificadores foram implementados, cada um empregando um grupo de curvas extraídas a partir das cotas obtidas para cada um dos cinco tipos de mapeamento. O classificador I se baseia em curvas principais extraídas a partir de cotas obtidas empregando-se o mapeamento do tipo I às vistas do conjunto de projeto. Da mesma forma, o classificador II se baseia em curvas principais extraídas a partir de cotas obtidas empregando-se o mapeamento do tipo II às vistas do conjunto de projeto e assim sucessivamente. A Tabela 4.2 apresenta o número de segmentos que compõem cada uma das curvas principais.

A extração das curvas principais foi realizada empregando-se os conjuntos de projeto descritos na Seção 4.2 como entrada para o algoritmo de k-segmentos não-suave [7], o qual foi apresentado na Seção 2.2.4. Todas as curvas foram extraídas atribuindo-se os seguintes valores aos parâmetros de entrada do algoritmo: $k\_max = 30$, $alfa = 1$ e $lambda = 1$. Note que se desejou obter curvas com o maior número possível de segmentos visto que, de acordo com os resultados apresentados no Item 3.3.3, a eficiência média de classificação aumenta diretamente com o número de segmentos das curvas representativas das classes. Como foi observado, o valor de k_max usado foi suficiente para permitir a extração da quantidade máxima de segmentos. O valor empregado para o parâmetro *alpha* foi arbitrário, visto que esse parâmetro não é considerado pelo algoritmo de k-segmentos não-suave no processo extração das curvas (mais especificamente, na obtenção das matrizes *vertices* e *edges* que caracterizam as curvas). O valor atribuído ao parâmetro *lambda* garantiu que a interligação dos segmentos considerasse, com igual importância, a minimização do comprimento final da curva e dos ângulos formados pelas junções de segmentos vizinhos.

Da análise da Tabela 4.2 verifica-se que a curva da classe I (porta-aviões) é a que possui a maior quantidade de segmentos. Também é possivel observar que as curvas extraídas empregando-se as cotas obtidas pelo mapeamento do tipo II possuem uma pequena quantidade de segmentos. Isto ocorre porque a forma do contorno lateral das vistas não varia muito em funçao dos ângulos de azimute e elevação, ao contrário do que ocorre, por exemplo, com o contorno superior. As-

Tabela 4.2: Número de segmentos que compõem as curvas principais das diversas classe extraídas a partir das cotas obtidas empregando-se os cinco métodos de mapeamento.

	Classe 1	**Classe 2**	**Classe 3**	**Classe 4**	**Classe 5**
Map. Tipo I	26	22	18	17	19
Map. Tipo II	6	2	8	5	1
Map. Tipo III	22	22	20	21	20
Map. Tipo IV	23	22	25	23	22
Map. Tipo V	23	19	23	19	21

sim, essas cotas compõem um conjunto compacto de pontos, os quais podem ser representados através de curvas que possuem poucos segmentos.

As cotas extraídas representam um ponto num espaço multidimensional (300 ou 400 dimensões, de acordo com o tipo de mapeamento utlizado). A classificação é realizada medindo-se a distância euclidiana do ponto representado pelo conjunto de cotas a cada um dos segmentos que compõem as curvas principais. A vista é atribuída à classe cuja curva representativa encontra-se na menor distância euclidiana do ponto representado pelas cotas.

A rotina que realiza a classificação foi desenvolvida, inicialmente, em Matlab6 e, em seguida, codificada em linguagem C. Os arquivos que descrevem os modelos das classes e as rotinas de mapeamento de vistas foram implementados em Matlab6. Os códigos-fonte desses programas são listados no Apêndice A.

4.5 Resultados

Nos itens seguintes são apresentadas as matrizes de confusão para os conjuntos de projeto e de teste correspondentes a cada um dos cinco classificadores, além de gráficos contendo a distribuição dos erros de classificação em função dos ângulos de elevação e azimute. Cada linha da matriz de confusão contém os porcentuais de classificação dos eventos de uma dada classe, permitindo que se identifique, para quais classes, um número maior de erros é cometido. A comparação dos resultados obtidos para os conjuntos de projeto e teste permite avaliar a capacidade de generalização do classificador. É desejável que a matriz de confusão seja o mais próxima possível de uma matriz diagonal. Neste caso, o classificador obtém a eficiência máxima,

classificando corretamente todos os eventos pertencentes às diversas classes.

Após a apresentação das matrizes de confusão, serão apresentados histogramas que mostram a distribuição dos erros de classificação em função dos valores dos ângulos de azimute e de elevação para os quais foram geradas as vistas. Intuitivamente, se espera que os erros de classificação se concentrem nas faixas de valores dos ângulos de azimute e elevação para as quais um operador encontra maior dificuldade em identificar um navio, o que ocorre, com maior freqüência, para valores do ângulo de azimute próximos a -90^o e $+90^o$, quando o navio é visto de proa ou de popa, e para os maiores ângulos de elevação. Além disso, é esperado que os padrões exibidos nos histogramas que mostram a distribuição dos erros em função do ângulo de azimute sejam razoavelmente simétricos com relação ao ângulo de 0^o, uma vez que para um operador, geralmente, encontra dificuldade equivalente na identificação um navio que se desloca, por exemplo, da direita para a esquerda ou no sentido oposto. Essa tendências, no entanto, não serão observadas para a maioria dos histogramas que serão mostrados. De fato, verifica-se que os histogramas que exibem os erros de classificação em função dos ângulos de azimute nem sempre apresentam padrões simétricos e, nem sempre, os erros são mais freqüentes para valores de ângulo de azimute próximos aos citados. Com relação aos padrões exibidos nos histogramas que mostram os erros em função dos ângulos de elevação das vistas, é possível afirmar que a maioria deles exibe um aspecto mais ou menos uniforme, indicando que o classificador aprende a classificar, aproximadamente, da mesma forma, vistas tomadas a qualquer ângulo de elevação.

No item 4.5.7 será apresentado um resumo dos resultados e será feita uma análise comparativa do desempenho de cada um dos classificadores, os quais foram designados de acordo com o tipo de mapeamento empregado na extração das cotas.

4.5.1 Classificador I

As Tabelas 4.3 e 4.4 apresentam as matrizes de confusão para o classificador I considerando, respectivamente, dados de projeto e de teste. Verifica-se uma eficiência média igual a 98,5% e 93,6%, respectivamente, para dados de projeto e de teste, o que pode ser considerado um bom resultado para a aplicação em pauta.

Conforme pode ser verificado da análise dos resultados apresentados, a classe

A é a que apresenta os menores porcentuais de acerto, tanto para dados de projeto (95,8%) como para dados de teste (88,6%). Verifica-se também que, apesar da quantidade de vistas pertencentes ao conjunto de projeto ser significativamente inferior à quantidade de vistas do conjunto de teste (o primeiro contém 144 vistas por classe enquanto que o segundo é composto por 8120 vistas por classe), o classificador I apresenta uma boa capacidade de generalização, visto que a diferença do porcentual de eficiência média para os dois conjuntos é pequena, menor que 5 pontos porcentuais.

Tabela 4.3: Matriz de confusão para o classificador I. Resultados para dados de *projeto* em porcentagem.

	Classe A	**Classe B**	**Classe C**	**Classe D**	**Classe E**
Classe A	**95,8**	2,1	2,1	0	0
Classe B	0	**100**	0	0	0
Classe C	0,7	0,7	**98,6**	0	0
Classe D	0,7	0	0,7	**97,9**	0,7
Classe E	0	0	0	0	**100**
Eficiência Média = 98,5					

Tabela 4.4: Matriz de confusão para o classificador I. Resultados para dados de *teste* em porcentagem.

	Classe A	**Classe B**	**Classe C**	**Classe D**	**Classe E**
Classe A	**88,6**	3,0	5,5	0,1	2,8
Classe B	1,0	**90,9**	0,9	0,1	7,1
Classe C	0,7	2,1	**95,1**	0,3	1,8
Classe D	0,4	0,2	1,1	**96,3**	2,0
Classe E	0	1,9	0,9	0,3	**96,9**
Eficiência Média = 93,6					

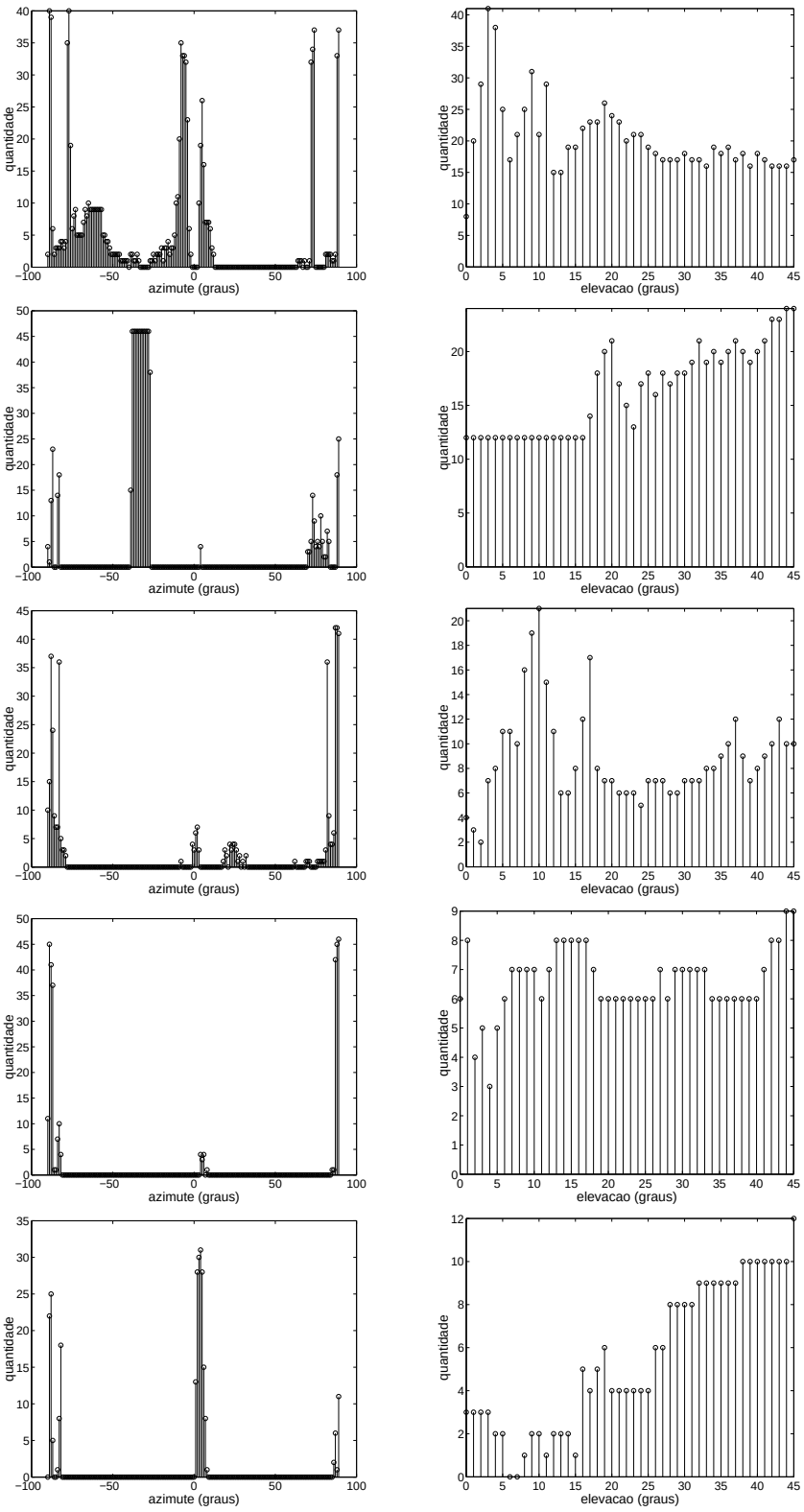


Figura 4.19: Histogramas de erros para o classificador I, para dados de teste, em função da classe (cl) e dos ângulos de azimute (az) ou elevação (el). Da esquerda para a direita e de cima para baixo: (a) cl A, az; (b) cl A, el; (c) cl B, az; (d) cl B, el; (e) cl C, az; (f) cl C, el; (g) cl D, az; (h) cl D, el; (i) cl E, az; (j) cl E, el.

A Figura 4.19 apresenta uma série de histogramas, os quais contém a distribuição dos erros de classificação para dados de teste. A primeira coluna contém os histogramas de erros em função do ângulo de azimute enquanto que a segunda contém os histogramas de erros em função do ângulo de elevação. A primeira linha contêm os histogramas de erros de classificação das vistas pertencentes à classe A; a segunda os histogramas referentes ao erros de classificação das vistas da classe B e assim sucessivamente. O histograma (a) mostra uma grande concentração de erros de classificação de vistas da classe A para ângulos de azimute negativos. Também são observados vários erros para ângulos positivos pequenos (entre 5^o e 15^o) e para ângulos entre $+75^o$ e $+89^o$. É possível que, incluindo-se no conjunto de projeto da classe A uma maior quantidade de vistas tomadas nessas faixas de ângulos de azimute, se possa melhorar a eficiência de classificação da classe. Da análise do histograma (b) verifica-se que as vistas da classe A classificadas incorretamente estão distribuídas, aproximadamente, de forma uniforme com relação aos ângulos de elevação. Com relação à classe B podemos perceber uma grande concentração de erros em vistas com ângulos de azimute entre $-45^o e -30^o$ (histograma (c)). Também podemos perceber que a quantidade de erros de classificação cresce a medida que aumenta o ângulo de elevação da vista (histograma (d)), fato também observado para vistas da classe C (histograma (f)). As vistas das classes C e D apresentam mais erros de classificação para ângulos de azimute grandes (histogramas (e) e (g)) enquanto que o oposto ocorre para as vistas da classe E (histograma (i)).

4.5.2 Classificador II

As Tabelas 4.5 e 4.6 apresentam as matrizes de confusão para o classificador II considerando, respectivamente, dados de projeto e de teste. Pode ser verificado que enquanto as classes A, C e D foram classificadas de forma razoável, os resultados obtidos para as classes B e E são inaceitáveis. Muito embora, inicialmente, não se considerasse possível obter bons resultados de classificação empregando o mapeamento do tipo II, o qual apenas considera parte da lateral das vistas, julgou-se importante verificar o quanto de informação discriminante se poderia extrair. De fato, conforme pode ser verificado da análise dos resultados apresentados, os valores de eficiência média de classificação não ultrapassam 50%, insatisfatórios para a

aplicação em questão.

Tabela 4.5: Matriz de confusão para o classificador II. Resultados para dados de *projeto* em porcentagem.

	Classe A	**Classe B**	**Classe C**	**Classe D**	**Classe E**
Classe A	**86,8**	0,7	8,3	2,8	1,4
Classe B	17,4	**12,5**	48,6	20,8	0,7
Classe C	18,1	0,7	**70,1**	9,7	1,4
Classe D	8,3	2,1	15,3	**74,3**	0
Classe E	22,2	21,5	43,8	11,8	**0,7**
Eficiência Média = 48,9					

Tabela 4.6: Matriz de confusão para o classificador II. Resultados para dados de *teste* em porcentagem.

	Classe A	**Classe B**	**Classe C**	**Classe D**	**Classe E**
Classe A	**76,8**	2,7	15,2	3,4	1,9
Classe B	21,1	**15,9**	40,9	19,5	2,6
Classe C	19,9	6,7	**55,6**	16,8	1,0
Classe D	12,7	2,3	19,5	**64,9**	0,6
Classe E	26,4	8,7	49,7	15,0	**0,2**
Eficiência Média = 42,7					

A Figura 4.20 apresenta os histogramas dos erros de classificação das vistas das cinco classes de navios, em função dos ângulos de azimute e elevação. Os erros de classificação para vistas das classes A, C e D são mais freqüentes para ângulos pequenos de azimute do que para ângulos próximos aos valores extremos, -90^o e $+89^o$, conforme verificado nos histogramas (a), (e) e (g). Com relação à distribuição dos erros para ângulos de elevação, verifica-se que a quantidade de erros aumenta para ângulos maiores (mais próximos de 45^o) no caso da classe A (histograma (b)). Os demais histogramas apresentam uma distribuição aproximadamente uniforme.

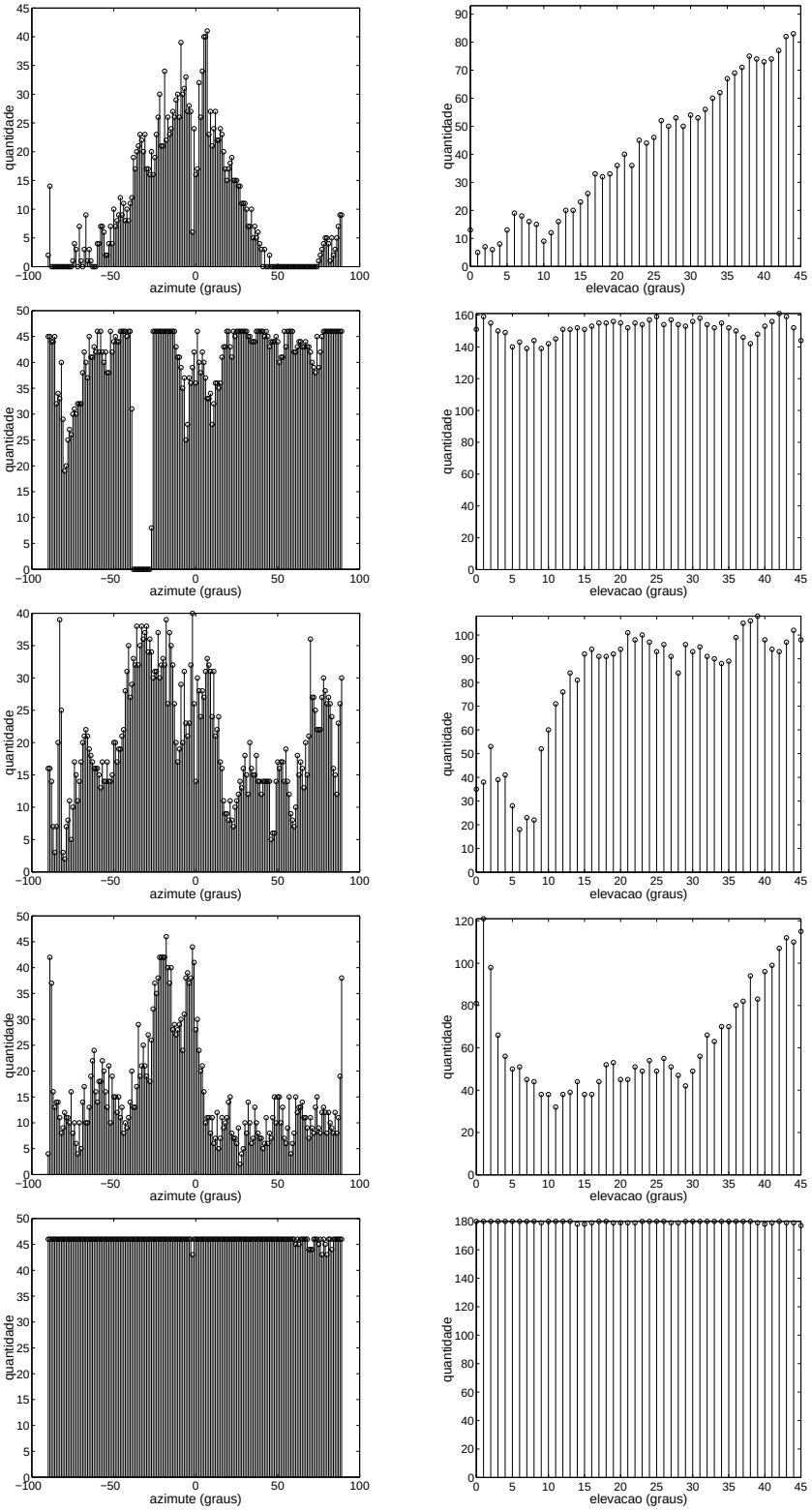


Figura 4.20: Histogramas de erros para o classificador II, para dados de teste, em função da classe (cl) e dos ângulos de azimute (az) ou elevação (el). Da esquerda para a direita e de cima para baixo: (a) cl A, az; (b) cl A, el; (c) cl B, az; (d) cl B, el; (e) cl C, az; (f) cl C, el; (g) cl D, az; (h) cl D, el; (i) cl E, az; (j) cl E, el.

4.5.3 Classificador III

As Tabelas 4.7 e 4.8 apresentam as eficiências de classificação das vistas das cinco classes para dados de projeto e teste, respectivamente. Verifica-se que a eficiência do classificador III, o qual possui curvas extraídas a partir de cotas extraídas conforme o mapeamento do tipo III, apresenta eficiências médias de classificação inferiores que as obtidas para o classificador I. A inclusão da informação de parte do contorno lateral das vistas (informação empregada pelo classificador II), a qual, conforme verificado no item 4.5.3 não possui boa capacidade de discriminação das classes, acarreta num ligeiro decréscimo no valor da eficiência do classificador III (97,4% e 92,9%, respectivamente, para dados de projeto e de teste) com relação ao classificador I (98,5% e 92,2%, respectivamente, para dados de projeto e de teste) o qual emprega, apenas, informações extraídas a partir da forma dos contornos superiores das vistas. A classe C apresenta um decréscimo considerável na sua eficiência que passa de 95,1%, para o classificador I, para 93,1%. Supõe-se que a adição das cotas laterais tenha tornado suas características mais semelhantes às de outras classes, acarretando na perda de eficiência observada. É interessante, no entanto, notar que a eficiência de classificação da classe A obtida com o classificador III (98,6% e 95,1%, respectivamente, para dados de projeto e de teste) é significativamente maior que à obtida com o emprego do classificador I (93,1% e 92,4%, respectivamente, para dados de projeto e de teste). Assim, dependendo da importância relativa desta classe, pode ser vantajoso optar pelo emprego do classificador III em lugar do classificador I.

Tabela 4.7: Matriz de confusão para o classificador III. Resultados para dados de *projeto* em porcentagem.

	Classe A	**Classe B**	**Classe C**	**Classe D**	**Classe E**
Classe A	**98,6**	0	0,7	0	0,7
Classe B	2,1	**96,5**	0	0	1,4
Classe C	3,5	2,1	**93,1**	0	1,4
Classe D	0,7	0	0	**99,3**	0
Classe E	0	0	0,7	0	**99,3**
Eficiência Média = 97,4					

Tabela 4.8: Matriz de confusão para o classificador III. Resultados para dados de *teste* em porcentagem.

	Classe A	**Classe B**	**Classe C**	**Classe D**	**Classe E**
Classe A	**92,4**	0,5	5,4	0,1	1,7
Classe B	0,8	**91,5**	4,8	0,2	2,7
Classe C	3,2	4,3	**89,2**	1,0	2,3
Classe D	0,3	0,4	1,7	**96,6**	1,0
Classe E	0,5	3,0	1,2	0,2	**95,1**
Eficiência Média = 92,9					

Os diversos histogramas da Figura 4.21 mostram a distribuição dos erros de classificação das vistas das cinco classes em função dos ângulos de azimute e elevação. Do histograma (a) verificamos que uma grande quantidade de vistas da classe A, com ângulos de azimute próximos de 0^o, -90^o e $+89^o$, são classificadas erroneamente, padrão também verificado para o classificador I (conforme mostrado no histograma (a) da Figura 4.19). O histograma (e) apresenta uma distribuição semelhante para vistas da classe C. As maiores concentrações de erros de classificação para as classes B e D ocorrem para ângulos extremos de azimute, próximos de -90^o e $+89^o$, conforme mostrado nos histogramas (c) e (g). A maior parte dos erros de classificação das vistas da classe E ocorrem para ângulos de azimute próximos a $+50^o$ e ângulos de elevação em torno de 40^o, conforme pode ser verificado pelos histogramas (i) e (j).

4.5.4 Classificador IV

As matrizes de confusão para o classificador IV estão contidas nas Tabelas 4.9 e 4.10, as quais contêm as eficiências de classificação para dados de projeto e teste, respectivamente. Como pode ser verificado, este tipo de classificador, o qual emprega as curvas principais extraídas empregando-se as cotas extraídas conforme o mapeamento do tipo IV, obteve uma eficiência para dados de projeto igual a 96,1% enquanto que para dados de teste foi atingida uma eficiência média igual a 94,5%, o melhor resultado para dados de teste, dentre os classificadores já apresentados.

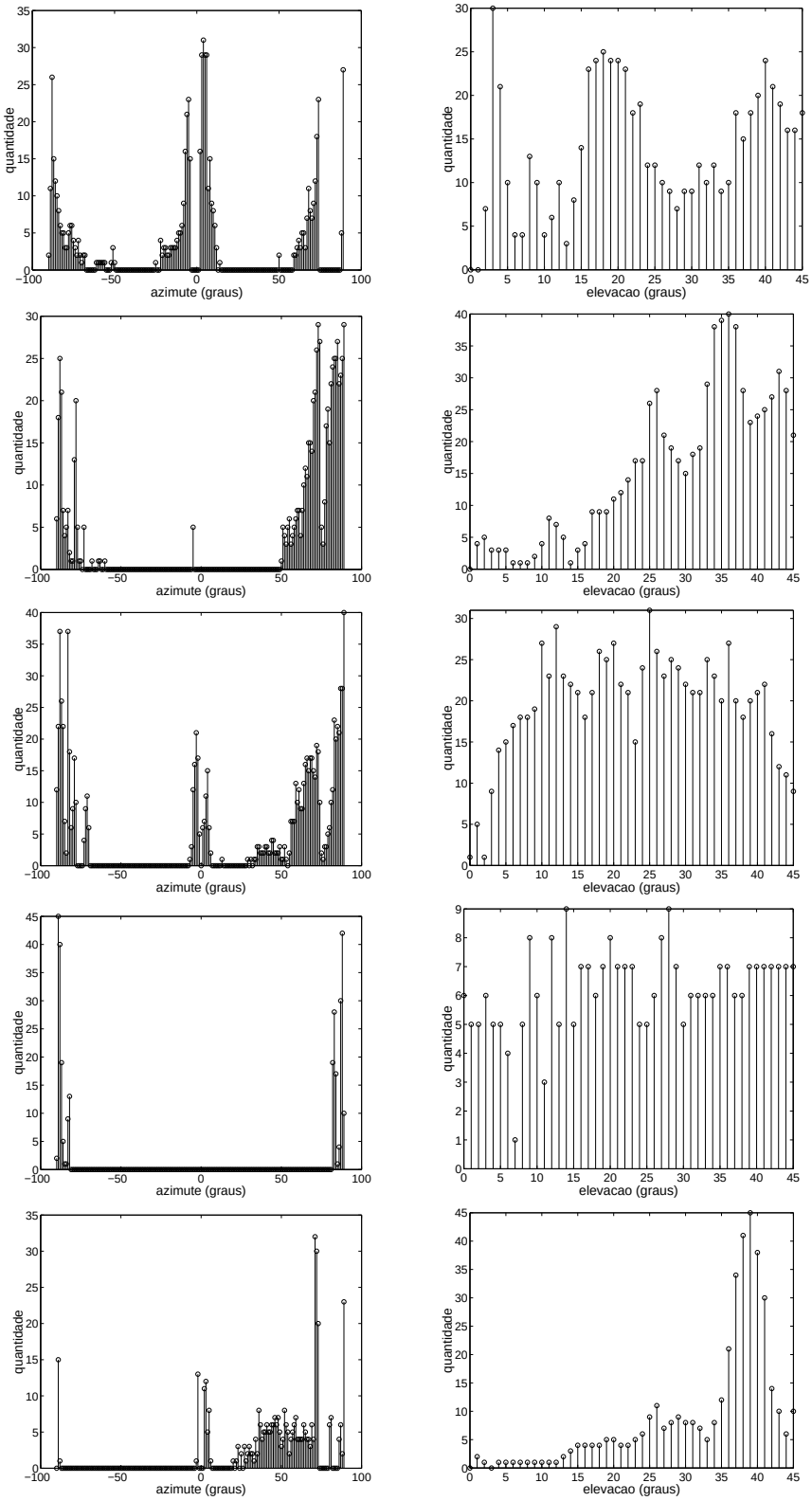


Figura 4.21: Histogramas de erros para o classificador III, para dados de teste, em função da classe (cl) e dos ângulos de azimute (az) ou elevação (el). Da esquerda para a direita e de cima para baixo: (a) cl A, az; (b) cl A, el; (c) cl B, az; (d) cl B, el; (e) cl C, az; (f) cl C, el; (g) cl D, az; (h) cl D, el; (i) cl E, az; (j) cl E, el.

Os resultados por classe também são bastante satisfatórios, tendo todas as classes alcançado porcentuais de classificação superiores a 90%. Outro ponto importante é a boa capacidade de generalização, evidenciada pela pequena diferença entre os resultados de eficiência alcançados para dados de projeto e de teste.

Tabela 4.9: Matriz de confusão para o classificador IV. Resultados para dados de *projeto* em porcentagem.

	Classe A	**Classe B**	**Classe C**	**Classe D**	**Classe E**
Classe A	**96,5**	1,4	0,7	0	1,4
Classe B	2,1	**95,8**	0	0,7	1,4
Classe C	0,7	0,7	**91,0**	0,7	6,9
Classe D	0	0	0,7	**99,3**	0
Classe E	0	0,7	1,4	0	**97,9**
Eficiência Média = 96,1					

Tabela 4.10: Matriz de confusão para o classificador IV. Resultados para dados de *teste* em porcentagem.

	Classe A	**Classe B**	**Classe C**	**Classe D**	**Classe E**
Classe A	**92,9**	4,7	1,1	0,4	1,0
Classe B	1,7	**94,5**	2,0	0,6	1,2
Classe C	0,7	1,8	**91,6**	1,3	4,6
Classe D	0,3	0,2	1,1	**97,7**	0,7
Classe E	1,8	0,6	1,8	0,1	**95,7**
Eficiência Média = 94,5					

A Figura 4.22 contém dez histogramas que mostram a distribuição dos erros de classificação das vistas pertencentes ao conjunto de teste, em função dos ângulos de azimute e elevação, para o classificador IV. Os histogramas (a) e (g) apresentam padrões similares, indicando a ausência de erros de classificação para vistas das classes A e D, com ângulos de azimute na faixa entre -50^o e $+50^o$. Os histogramas (e) e (i), referentes às classes C e E, também apresentam padrões similares, indicando

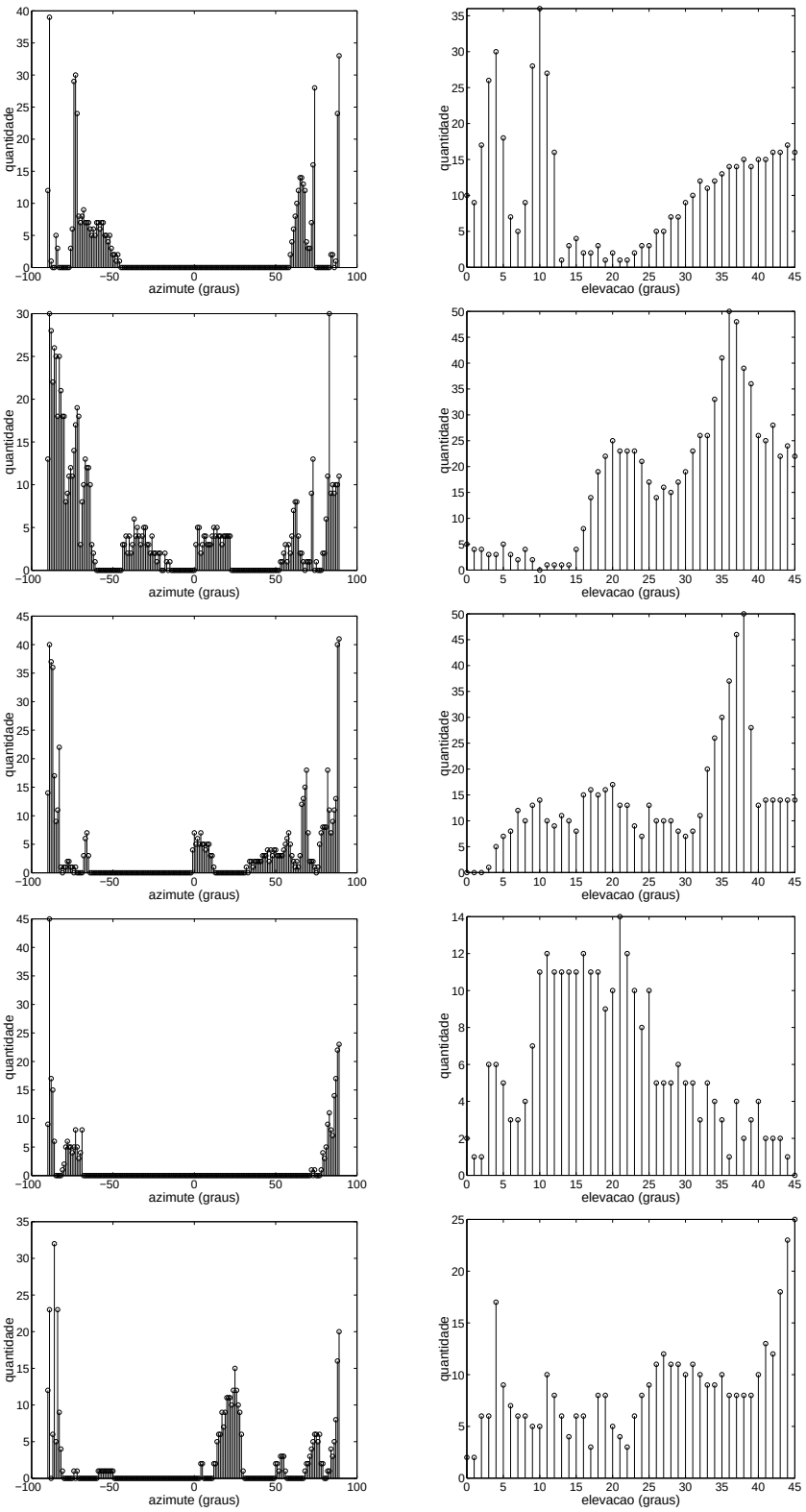


Figura 4.22: Histogramas de erros para o classificador IV, para dados de teste, em função da classe (cl) e dos ângulos de azimute (az) ou elevação (el). Da esquerda para a direita e de cima para baixo: (a) cl A, az; (b) cl A, el; (c) cl B, az; (d) cl B, el; (e) cl C, az; (f) cl C, el; (g) cl D, az; (h) cl D, el; (i) cl E, az; (j) cl E, el.

a presença de erros de classificação em toda a faixa de ângulos de azimute exceto no intervalo entre -50^o e 0^o. Os erros de classificação de vistas da classe B estão distribuídos em toda a faixa de ângulos de azimute, estando, no entanto, mais concentrados no intervalo entre -90^o e -60^o. Com relação à distribuição de erros em função do ângulo de elevação, não há um padrão dominante, estando os erros, na maioria dos casos, aproximadamente, uniformemente distribuídos, exceto a classe B a qual só passa a apresentar muitos erros de classificação para ângulos maiores que 15^o.

4.5.5 Classificador V

As matrizes de confusão para o classificador V são apresentadas nas Tabelas 4.11 e 4.12, as quais se referem a dados de projeto e de teste, respectivamente. Como pode ser observado, os resultados alcançados superam as eficiências médias obtidas para o classificador IV. Esse resultado mostra que o emprego de cotas internas, tomadas a partir do centro de massa da vista, permite a extração de características que possuem um potencial de discriminação das classes melhor que o obtido para os outros tipos de mapeamento empregados. Aparentemente, a obtenção de cotas a partir de ponto calculado a partir das coordenadas dos pontos de contorno, como o centro de massa, possibilitou a extração de cotas contendo um maior potencial de discriminação.

Para dados de projeto foi obtida uma eficiência média igual a 98,8% enquanto que para dados de teste a eficiência média foi igual a 97,3%, resultados bastante expressivos. Considerando dados de teste, obteve-se, para todas as classes, eficiências de classificação maiores que 93% (a menor eficiência foi obtida para a classe B, a qual foi igual a 93,7%).

Os histogramas mostrados na Figura 4.23 mostram a distribuição de erros de classificação das vistas pelo classificador V. Os histogramas (a), (e), (g) e (i) mostram uma concentração de erros de classificação de vistas tomadas para ângulos de azimute próximos de -90^o e $+89^o$, padrão já verificado para outros classificadores. O histograma (c), relativo aos erros de classificação de vistas da classe B, no entanto, mostra uma quantidade significativa de erros de classificação para ângulos de azimute em toda a faixa, exceto para ângulos entre $+20^o$ a $+70^o$, onde só se observam erros

Tabela 4.11: Matriz de confusão para o classificador V. Resultados para dados de *projeto* em porcentagem.

	Classe A	**Classe B**	**Classe C**	**Classe D**	**Classe E**
Classe A	**98,6**	0	0,7	0,7	0
Classe B	0	**97,2**	2,1	0,7	0
Classe C	0	0,7	**99,3**	0	0
Classe D	0	0,7	0,7	**98,6**	0
Classe E	0	0	0	0	**100**
Eficiência Média = 98,8					

Tabela 4.12: Matriz de confusão para o classificador V. Resultados para dados de *teste* em porcentagem.

	Classe A	**Classe B**	**Classe C**	**Classe D**	**Classe E**
Classe A	**98,0**	0,3	1,1	0,4	0,2
Classe B	2,7	**93,7**	1,9	0,2	1,4
Classe C	0,7	1,0	**97,2**	0,4	0,7
Classe D	0,1	0,2	0,3	**98,8**	0,6
Classe E	0,1	0,6	0,6	0	**98,7**
Eficiência Média = 97,3					

para vistas tomadas com ângulos de azimute próximos de $+50^o$. Os histogramas (d), (f) e (j) mostram que as vistas das classes B, C e E, geradas considerando-se ângulos de elevação pequenos, são classificadas corretamente, só se verificando erros para ângulos maiores que 6^o. Os histogramas correspondentes às classes A e D, (b) e (h), possuem padrões semelhantes, mostrando que a maior parte dos erros ocorre para vistas geradas para ângulos pequenos de elevação.

Note que na maioria dos tipos de mapeamento propostos evitou-se extrair informação discriminante da parte de baixo da vista, o que, no caso em questão, corresponde à interface entre o costado do navio e a superfície do mar porque não há informação relevante neste trecho. Como no estudo em tela se empregam vistas sintéticas, este trecho, apesar de não contribuir com informação relevante, não introduz ruído, o que pode ocorrer quando for empregado um conjunto de dados real, onde se verificam sombras junto ao navio, principalmente devido à sua esteira (traço

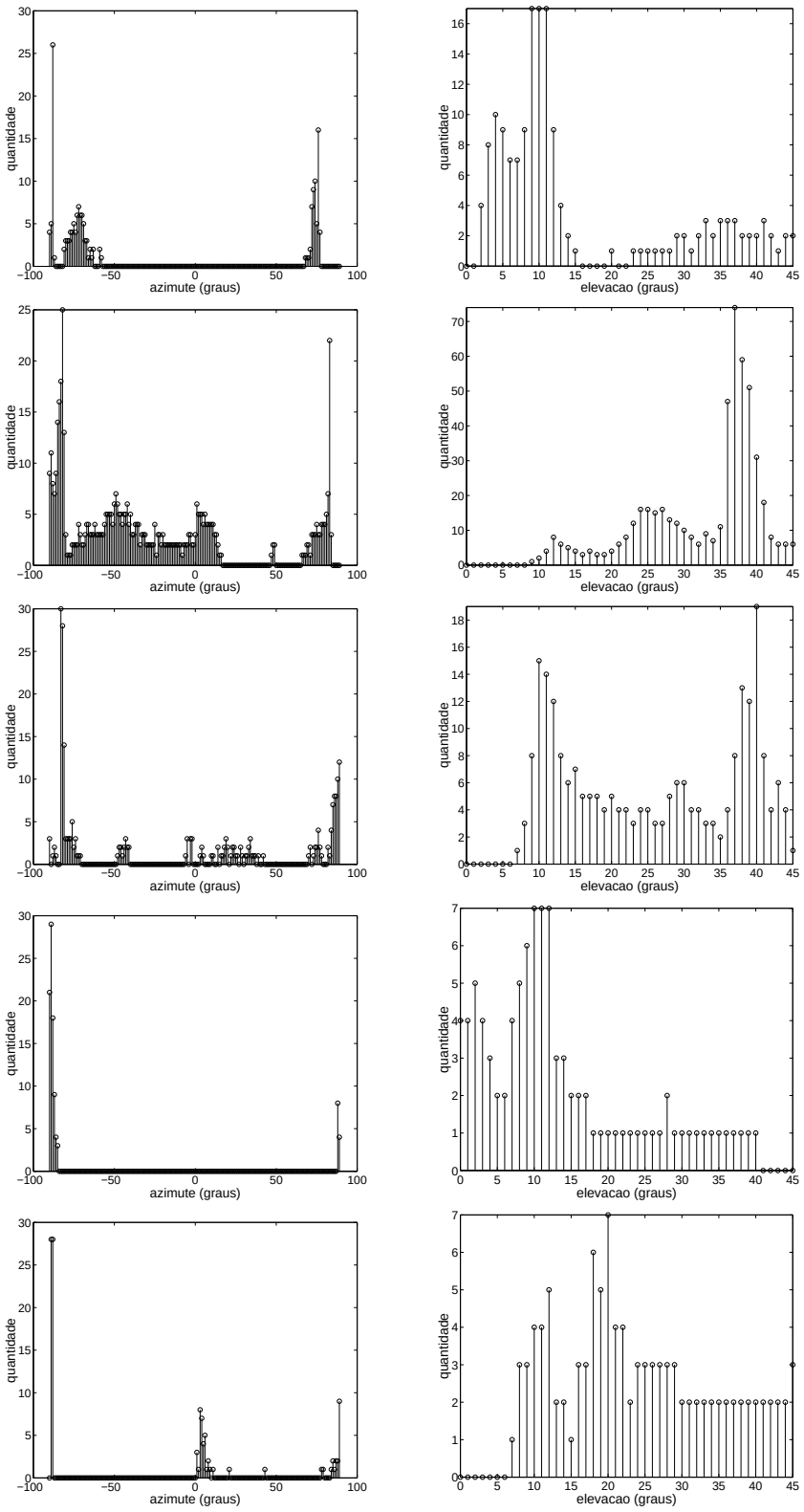


Figura 4.23: Histogramas de erros para o classificador V, para dados de teste, em função da classe (cl) e dos ângulos de azimute (az) ou elevação (el). Da esquerda para a direita e de cima para baixo: (a) cl A, az; (b) cl A, el; (c) cl B, az; (d) cl B, el; (e) cl C, az; (f) cl C, el; (g) cl D, az; (h) cl D, el; (i) cl E, az; (j) cl E, el.

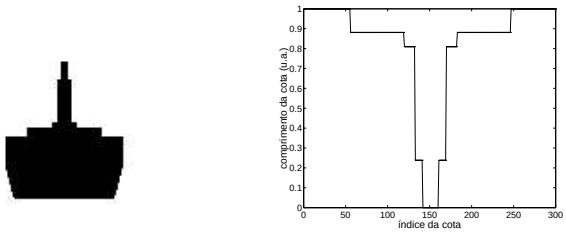
deixado na superfície do mar devido ao deslocamento de um navio) a qual pode ser interpretada, erroneamente, pelo algoritmo de segmentação.

4.5.6 Análise do desempenho do classificador para dados ruidosos

A fim de verificar o comportamento dos classificadores para dados de entrada ruidosos, adicionou-se ruído gaussiano, com diferentes níveis de energia, às cotas obtidas conforme os diversos métodos de mapeamento. As curvas principais extraídas a partir dos conjuntos de projeto originais, as quais são parte integrante dos sistemas classificadores, foram mantidas inalteradas.

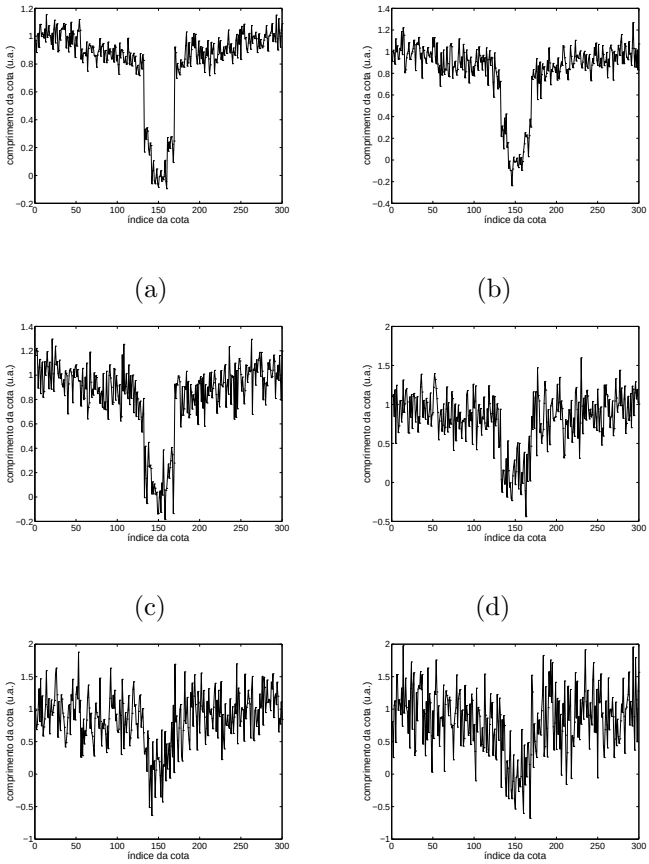
A Figura 4.24 mostra uma vista arbitrária da classe C e os valores das cotas extraídas dessa vista empregando-se o método I. Note que as cotas são normalizadas, possuindo valores compreendidos no intervalo $[0, 1]$. A Figura 4.25 mostra o resultado da adição de ruído às cotas mostradas na Figura 4.24 (b). Seis níveis de ruído foram empregados: 0,5% ($\sigma_n^2 = 0,005$), 1% ($\sigma_n^2 = 0,01$), 2% ($\sigma_n^2 = 0,02$), 5% ($\sigma_n^2 = 0,05$), 10% ($\sigma_n^2 = 0,1$) e 20% ($\sigma_n^2 = 0,2$). Não foram empregados níveis de ruído maiores que 20% porque se considera que o nível de ruído presente em imagens reais segmentadas não deverá exceder este patamar. Para cada uma das cinco classes e cada um dos cinco métodos de extração de cotas, seis conjuntos de teste foram gerados, adicionado-se ruído aos valores das cotas.

Os resultados de classificação para esses dados são mostrados na Tabela 4.13. É possível observar que nenhum classificador colapsou frente a dados ruidosos. Além disso, pode-se observar que, desconsiderando o classificador II, devido a sua ineficiência, os demais são razoavelmente robustos com relação à operação para dados contaminados com ruído aditivo gaussiano. O classificador V mantém-se o mais eficiente para níveis de ruído de até 2%, sendo então sobrepujado pelo classificador III, o qual, mantém-se o mais eficiente para os demais níveis de ruído empregados. De fato, este classificador é bastante robusto, mantendo uma boa eficiência média mesmo para dados muito ruidosos. A Figura 4.26 mostra, graficamente, a variação da eficiência média de cada classificador em função do nível do ruído adicionado aos dados de teste.



(a) (b)

Figura 4.24: Dados sem ruído: (a) vista arbitrária da classe C e (b) cotas extraídas dessa vista segundo o mapeamento do tipo 1.



(a) (b)

(c) (d)

(e) (f)

Figura 4.25: Resultado da adição de ruído gaussiano com valores crescentes de variância às cotas mostradas no gráfico da Figura 4.24(b): (a) $\sigma^2 = 0,005$; (b) $\sigma^2 = 0,01$; (c) $\sigma^2 = 0,02$; (d) $\sigma^2 = 0,05$; (e) $\sigma^2 = 0,1$; (f) $\sigma^2 = 0,2$.

4.5.7 Análise comparativa do desempenho dos classificadores e considerações práticas

A Tabela 4.14 resume os resultados de eficiência dos cinco tipos de classificadores apresentados neste capítulo. As duas primeiras linhas contêm as eficiências

Tabela 4.13: Comparação do desempenho de cada tipo de classificador para dados de teste ruidosos. Valores de variância do ruído (σ_n^2) em unidades arbitrárias (u.a.) e de eficiência média de classificação em porcentual (%).

σ^2	**Classif. 1**	**Classif. 2**	**Classif. 3**	**Classif. 4**	**Classif. 5**
0	93,6	42,7	92,9	93,6	97,3
0,005	93,4	41,7	92,8	93,2	97,0
0,01	93,0	40,9	92,6	92,0	96,6
0,02	92,3	40,0	92,3	89,8	95,5
0,05	89,1	37,6	91,1	85,4	90,9
0,1	84,4	35,8	89,1	79,8	84,0
0,2	77,3	33,6	85,4	72,8	73,1

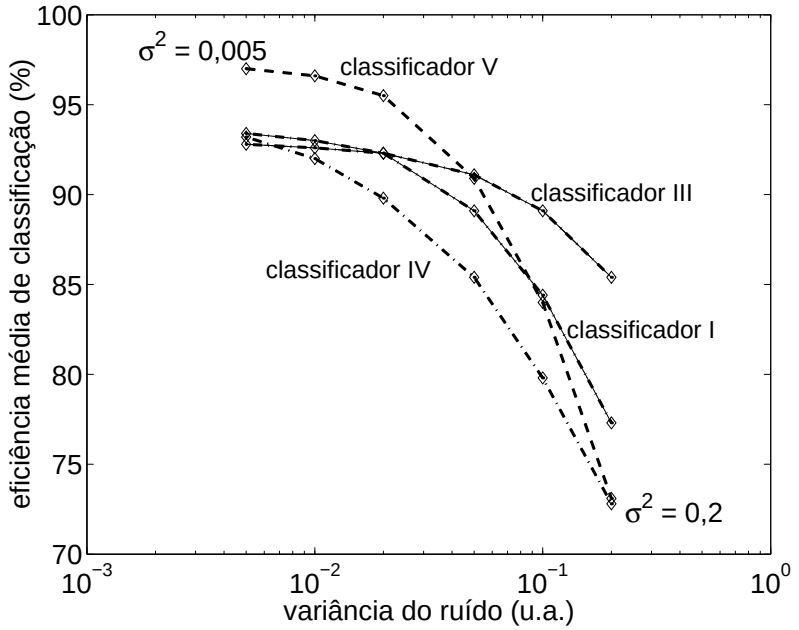


Figura 4.26: Eficiência média dos classificadores Tipo I, III, IV e V para dados de teste aos quais se adicionou ruído gaussiano com valores de variância na faixa compreendida entre 0,005 e 0,2.

médias para dados de teste, sem adição de ruído. É possível verificar que o classificador V é o mais eficiente, identificando corretamente 97,3% das vistas pertencentes aos conjuntos de teste, resultado significativamente superior ao obtido em [78] empregando-se, para os mesmos dados, um classificador neural baseado em momentos invariantes das vistas [85], o qual obteve uma eficiência média de classificação igual a 87,3%.

Verificou-se que os classificadores propostos são robustos com relação à adição

Tabela 4.14: Comparação do desempenho de cada tipo de classificador para dados de projeto e teste. Valores de eficiência média de classificação em porcentual (%).

Dados de Entrada	***Classif. Tipo I***	***Classif. Tipo II***	***Classif. Tipo III***	***Classif. Tipo IV***	***Classif. Tipo V***
Projeto	98,5	48,9	97,3	95,6	98,7
Teste sem ruído	93,6	42,7	92,9	93,6	97,3
Teste com ruído, $\sigma_n^2 = 0,1$	84,4	33,6	89,1	79,8	84,0
Teste com ruído, $\sigma_n^2 = 0,2$	77,3	33,6	85,4	72,8	73,1

de ruído gaussiano. O classificador V, por exemplo, obteve uma eficiência média de classificação melhor que 90% para dados de teste aos quais foi adicionado ruído gaussiano com variância $\sigma_n^2 = 0,05$, o que corresponde a 5% da amplitude máxima das cotas que compõem o conjunto de dados. Para um nível maior de ruído, igual a 20%, o classificador III apresenta a maior eficiência média, cujo valor é superior a 85%.

Conforme foi verificado por meio dos resultados de eficiência apresentados, é possível obter um classificador de vistas de navios, baseado em curvas principais, que seja eficiente. É importante, no entanto, que algumas questões práticas sejam abordadas. O classificador proposto emprega exclusivamente informação discriminante obtida a partir da forma do contorno externo da vista do navio. A motivação ao empregar este tipo de informação foi intuitiva: quando um navio surge no horizonte não é possível perceber mais detalhes que a forma dos mastros visíveis e esta informação, na maioria dos casos, é suficiente para que se identifique sua classe. Além disso, o classificador proposto se apoia num sistema composto por sensores infravermelhos, os quais não produzem imagens muito detalhadas. Ao se empregar apenas informações extraídas a partir dos contornos externos das vistas também se desejou simplificar o projeto do classificador, empregando um algoritmo de segmentação mais simples.

O classificador proposto, no entanto, não foi testado empregando-se imagens segmentadas reais, visto que um banco de dados deste tipo não estava disponível. Sabe-se que os classificadores são relativamente insensíveis a ruídos inseridos durante o processo de segmentação das imagens dos navios. No entanto, diversos problemas

podem ocorrer durante o processo de segmentação, tais como a perda de parte do contorno do navio (o qual pode ser confundido com o meio) e a inclusão da sombra do navio ou da sua esteira (traço deixado pelo navio, na superfície do mar, durante seu deslocamento) na imagem segmentada. É necessário que esses aspectos e outros que possam surgir durante a fase de investigação empregando um banco de dados reais sejam analisados a fim de determinar-se a robustez do classificador. A análise apresentada, baseada na adição de ruído às cotas extraídas a partir das vistas sintéticas, constitui-se numa análise preliminar.

O classificador baseado em curvas principais e informações de contorno das vistas possui diversas características desejáveis:

1. É facilmente escalável, ou seja, novas classes podem ser incorporadas bastando que se incluam as suas respectivas curvas principais;

2. Sua configuração é simples, bastando que se incluam ou removam curvas a fim de incluir ou remover classes;

3. Possui baixa complexidade computacional, $O(kn)$, onde k é a quantidade de segmentos das curvas principais empregadas e n é a dimensão dos dados.

4.6 Sumário

Neste Capítulo abordou-se o projeto de um classificador de navios baseado em curvas principais, o qual emprega informações extraídas do contorno externo das imagens de navios como informação discriminante das classes envolvidas. O classificador proposto pode ser empregado em equipamentos militares de defesa de uma plataforma ou em um sistema de apoio na identificação de um navio a partir de imagens infravermelhas.

O classificador recebe como entrada uma imagem segmentada de uma navio, extrai informação discriminante a partir do contorno da imagem e, empregando um conjunto de curvas principais, as quais caracterizam as classes envolvidas, identifica sua classe com uma probabilidade de acerto aceitável.

Cinco classes distintas de navios – porta-aviões, contratorpedeiro, fragata, navio-tanque e navio-patrulha – foram modeladas empregando-se os recursos dis-

poníveis no Toolbox de Visualização 3D do *Matlab*©5.3.0. A partir desses modelos foram geradas, para cada classe, 8120 vistas, considerando diversos ângulos de azimute e elevação.

Cinco métodos de extração de informação discriminante foram propostos e empregados na obtenção de cotas que mapearam a contorno externo das vistas. Para cada classe de navio e cada tipo de mapeamento, 144 vistas selecionadas foram empregadas na obtenção das cotas que constituiram os conjuntos de projeto, os quais foram empregados como entrada para o algoritmo de k-segmentos não-suave [7], usado na extração das curvas principais. A divisão das cotas em conjuntos de projeto e de teste seguiu o mesmo procedimento adotado por Amaral [78] o qual, usando os mesmos dados, propôs um classificador neural de imagens segmentadas de navios baseado em momentos invariantes [85]. Desta forma, foi possível comparar os resultados obtidos neste trabalho com os obtidos em [78].

O melhor classificador obteve, para dados de teste, uma eficiência média igual a 97,3%, um resultado significativamente melhor que o obtido empregando-se o classificador neural proposto em [78] (87,3%). Além disso, conforme discutido no Item 4.5.7, o classificador baseado em curvas principais possui várias características atraentes tais como sua escalabilidade, a facilidade de configuração, além do baixo custo computacional durante a fase de operação, característica que também pode ser atribuída ao classificador neural.

Também foi realizada uma análise do desempenho do classificador para dados ruidosos tendo sido obtida uma eficiência média igual a 89,1% para um nível de ruído Gaussiano aditivo igual a 10% e 85,4% para um nível de ruído igual a 20%, resultados que mostram que o classificador baseado em curvas principais é razoavelmente robusto a este tipo de ruído. A fim de avaliar a robustez do classificador é necessário que se considerem diversos efeitos que podem ocorrer durante o processo de segmentação de uma imagem real, os quais não foram considerados na análise preliminar apresentada.

Capítulo 5

Conclusão

Este trabalho explora o emprego de curvas principais como uma ferramenta de processamento estatístico, usada no desenvolvimento de dois classificadores de navios, visando seu emprego em equipamentos de defesa da Marinha do Brasil.

Inicialmente, foi apresentada uma revisão de curvas principais, as quais podem ser vistas como uma generalização da técnica de análise de componentes principais. Essa ferramenta visa, basicamente, a representação compacta de dados multimensionais e, neste trabalho, foi empregada na extração de assinaturas de navios, as quais, agrupadas sob a forma de um banco de dados, formam a base dos classificadores apresentados.

Foram abordados os conceitos fundamentais introduzidos por Hastie e Stutzle [1] e comentados os problemas potenciais envolvendo a extração de curvas principais para conjuntos de dados finitos. Em seguida, são abordados diversos trabalhos subseqüentes, os quais aprimoram a definição e os métodos inicialmente propostos, apresentam novas definições e propõem algoritmos de extração alternativos. O algoritmo de k-segmentos não-suave [7], empregado na extração das curvas usadas no desenvolvimento dos classificadores propostos, é abordado em detalhe. Concluindo a discussão sobre este primeiro tópico, alguns algoritmos de extração de curvas principais apresentados são comparados.

Em seguida, foi desenvolvido um classificador de navios utilizando informação espectral extraída a partir de sinais de sonar passivo. Este classificador foi implementado baseado na técnica de curvas principais. O conjunto de dados empregado consistiu em espectros obtidos a partir do pré-processamento adequado dos ruídos

detectados [6]. Três bancos de dados foram construídos a partir dos espectros: o primeiro consistiu dos próprios espectros; o segundo dos espectros normalizados; e o terceiro de espectros médios, obtidos extraindo-se os valores médios dos tons presentes em dois espectros sucessivos (em termos temporais) normalizados. Através de um conjunto de curvas extraídas para cada classe, e utilizando um critério simples de decisão, foi possível obter eficiências superiores a 85% (classe H), totalizando, para o conjunto de teste, uma eficiência média de classificação igual a 92,4%. Este valor de eficiência foi menor que o obtido empregando-se um classificador neural, cuja eficiência média variou no intervalo de 93,4% a 96,0%. No entanto, apesar do classificador baseado em curvas principais ter apresentado um valor de eficiência menor que o do classificador neural, a arquitetura do classificador baseado em curvas lhe confere características interessantes para a aplicação naval, tais como a facilidade de inclusão e de remoção de classes. Empregando-se não mais espectros singulares, mas a média de dois espectros sucessivos (método AN3) foi possível obter uma eficiência média de classificação igual a 96,7%, resultado bastante significativo, considerando-se a complexidade do problema. Uma análise das medidas das distâncias entre as curvas principais resultou útil na discussão dos resultados de classificação, mas não possibilitou uma análise da robustez do classificador, o que será sugerido como um possível trabalho futuro.

A segunda aplicação desenvolvida consistiu num classificador de navios, baseado em curvas principais, o qual emprega informações extraídas do contorno externo das imagens de navios, como informação discriminante das classes envolvidas. O classificador recebe como entrada uma imagem segmentada de um navio, extrai informação discriminante a partir deste contorno e, empregando um conjunto de curvas principais, as quais caracterizam as classes envolvidas, identifica sua classe. Cinco classes distintas de navios – porta-aviões, contratorpedeiro, fragata, navio-tanque e navio-patrulha – foram modeladas por Amaral [78]. A partir desses modelos foram geradas, para cada classe, 8120 vistas, considerando-se diversos ângulos de azimute e elevação. Cinco métodos de extração de informação discriminante foram propostos e empregados na obtenção de cotas que mapearam a contorno externo das vistas. O melhor classificador obteve, para dados de teste, uma eficiência média igual a 97,3%, um resultado significativamente melhor que o obtido empregando-se o classificador

neural proposto em [78] (87,3%). Além disso, tal como o classificador de sinais de sonar passivo proposto no Capítulo 3, este classificador possui várias características próprias que são adequadas ao seu emprego pelos meios navais.

Dois dos principais méritos dos classificadores apresentados são a sua escalabilidade e a sua facilidade de configuração. Através da extração de assinaturas acústicas, no caso do problema de classificação de sinais de sonar, ou de assinaturas infravermelhas de navios, no caso do classificador baseado em imagens infravermelhas segmentadas, realizada por meio do emprego de curvas principais, é possível incorporar novas classes ou remover classes indesejadas de forma rápida e simples.

Tendo em vista os bons resultados obtidos com o emprego de curvas principais na extração de assinaturas, concluímos que essa técnica é uma ferramenta adequada à solução de problemas complexos de representação e de classificação, tais como os focalizados neste trabalho.

5.1 Possíveis Trabalhos Futuros

O trabalho desenvolvido permite desdobramentos diversos, tais como:

1. **Alternativas na extração de curvas principais:** neste trabalho, o algoritmo de k-segmentos não suave foi utilizado na extração das curvas principais empregadas no desenvolvimento dos classificadores apresentados. Entretanto, diversos outros algoritmos foram propostos, implementados e disponibilizados por seus autores, podendo ser empregados na extração de novas curvas principais que podem substituir as curvas extraídas neste trabalho com o emprego do algoritmo de k-segmentos não-suave. Dentre eles podemos citar o PLA, implementado em Java, e disponível na página de Balázs Kégl, *www.iro.umontreal.ca/ ∼ kegl/research/pcurves*; o PCOP, implementado em C^{++}, disponibilizado mediante pedido ao autor, Pedro Delicado, *pedro.delicado@upc.edu*; e o algoritmo de k-segmentos suave, implementado em Matlab, e disponível na página de Jacob Verbeek, *http://staff.science.uva.nl/ ∼ verbeek/software/index-en.html*.

2. **Critério de classificação:** empregou-se como critério de classificação adotar a classe cuja distância euclidiana do espectro de entrada à sua curva repre-

sentativa é a menor dentre todas. Apesar dos bons resultados obtidos, é interessante que outras metodologias sejam analisadas. Sugere-se, por exemplo, que, inicialmente, se reduza a dimensão dos dados, através do emprego uma técnica adequada, como a de componentes principais de discriminação [3] ou de componentes principais não-lineares [4]. Em seguida, sugere-se a extração das curvas principais e a medição das distâncias de Mahalanobis entre o dado de entrada e os agrupamentos formados pelos dados de teste associados a cada segmento das curvas principais extraídas, atribuindo esse dado de entrada à classe do grupamento de dados mais próximo dele.

3. **Reconhecimento e inclusão de uma nova classe:** é interessante verificar a viabilidade de detecção automática de uma nova classe, inicialmente, apenas baseando-se na distância euclidiana do espectro às curvas representativas das classes já existente. É interessante, também que, após detectada uma nova classe, se considere a possibilidade de análise dos agrupamentos dos dados, visando uma nova extração de todas ou de algumas curvas (retreino do classificador).

4. **Integração do classificador de sinais de sonar passivo ao módulo de pré-processamento:** o classificador desenvolvido em linguagem C pode, facilmente, ser integrado ao módulo de pré-processamento desenvolvido por Soares-Filho [6]. Conforme discutido, o tempo de classificação de um espectro, considerando a implementação implementada em linguagem C, ambiente Linux, rodando num PC Athlon 2600XP com 512 MB de memória, em ambiente Linux, é igual a 2,8 ms, o que pode ser considerado um tempo curto, visto que a janela de amostragem do sinal possui uma largura igual a 186 ms. A implementação do sistema classificador poderia ser realizada em uma arquitetura baseada num PC ou em uma arquitetura híbrida, composta de um processador digital de sinais (DSP) e um PC atuando como *host*.

5. **Integração do classificador de imagens ao módulo de detecção e segmentação:** o classificador de imagens segmentadas, apresentado no capítulo 4, foi desenvolvido visando sua integração ao algoritmo de detecção de navios e segmentação de imagens proposto por Neves, da Silva e Mendonça [87], a

fim de obter-se um módulo de detecção e classificação automática de navios. É necessário codificar as rotinas que compõem esse módulo (essas rotinas, atualmente, estão codificadas em Matlab), integrá-las ao código do classificador (já implementado em linguagem C) e instalá-las no hardware escolhido, verificando as restrições de tempo de processamento impostas pela aplicação.

6. **Avaliação do classificador de navios empregando um banco de imagens infravermelhas reais:** devido à complexidade envolvida na tarefa, o módulo de segmentação de uma imagem real de um navio pode produzir um cotorno impreciso. Alguns dos efeitos possíveis são a inserção da esteira do navio, a perda de regiões do navio ou a inclusão de áreas preenchidas com caracteres sintéticos. Estes efeitos prejudicam a qualidade do contorno extraído, podendo causar uma redução da eficiência do classificador ou até mesmo o seu colapso.

7. **Análise de ruído e branqueamento dos dados de sonar passivo:** é interessante que amostras do ruído ambiente sejam disponibilizadas e analisadas, permitindo que se acrescentasse à cadeia de pré-processamento, o branqueamento dos dados [45].

8. **Avaliação da energia das curvas principais:** de forma similar à avaliação da energia extraída por cada componente na análise de componentes principais [44], sugere-se que se obtenha a energia associada a uma curva principal.

Referências Bibliográficas

[1] HASTIE, T. J., STUETZLE, W., "Principal curves", *Journal of the American Statistical Association*, v. 84, n. 406, pp. 502–516, 1989.

[2] BANFIELD, J., RAFTERY, A., "Ice flow identification in satellite images using mathematical morphology and clustering about principal curves", *Journal of the American Statistical Association*, v. 87, n. 417, pp. 7–16, 1992.

[3] DUDA, R. O., HART, P., STORK, D., *Pattern Classification, 2nd ed.* Willey-Interscience, 2002.

[4] KRAMER, M. A., "Nonlinear principal component analysis using autoassociative neural networks", *Journal of the American Institute of Chemical Engineering*, v. 37, n. 2, pp. 233–243, 1991.

[5] HYVÄRINEN, A., KARHUNEN, J., OJA, E., *Independent Component Analysis*. John Wiley and Sons, 2001.

[6] SOARES-FILHO, W., *Classificação do ruído irradiado por navios usando redes neurais*. Phd thesis, Universidade Federal do Rio de Janeiro, COPPE/UFRJ, 2001.

[7] VERBEEK, J. J., VLASSIS, N., KRÖSE, B., *A k-segments algorithm for finding principal curves*, Report, Intelligent Autonomous Systems Technical Report Series, nr. IAS-UVA-00-11, 2000.

[8] HASTIE, T. J., *Principal curves and surfaces*. Phd thesis, Stanford University, 1984.

[9] KéGL, B., *Principal curves: learning, design and applications*. Phd thesis, Concordia University, 1999.

[10] OLIVA, W. M., *Vetores e Geometria*. Edgard Blücher, 1973.

[11] CLEVELAND, W. S., DEVLIN, S. J., "Locally weighted regression: an approach to regression analysis by local fitting", *Journal of the American Statistical Association*, pp. 596–610, 1988.

[12] CLEVELAND, W. S., "Robust locally weighted regression and smoothing scatterplots", *Journal of the American Statistical Association*, v. 74, pp. 829–836, 1979.

[13] SILVERMAN, B. W., "Some aspects of spline smoothing regression: an approach to regression analysis by local fitting", *Journal of the American Statistical Association*, pp. 596–610, 1988.

[14] DUCHAMP, T., STUETZLE, W., "Extremal properties of principal curves in the plane", *Annals of Statistics*, v. 24, n. 4, pp. 1511–1520, 1995.

[15] DUCHAMP, T., STUETZLE, W., "Geometric properties of principal curves in the plane", *Robust statistics, data analysis and computer intensive methods: in honor of Petr Huber's 60th birthday*, v. 109 of Lecture Notes in Statistics, pp. 135–152, 1996.

[16] TIBSHIRANI, R., "Principal curves revisited", *Statistics and Computation*, v. 2, pp. 183–190, 1992.

[17] CHANG, K., GHOSH, J., "Probabilistic principal surfaces", *Proceedings of the International Joint Conference on Neural Networks*, p. 605, 1999.

[18] CHANG, K., GHOSH, J., "A unified model for probabilistic principal surfaces", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 23, n. 1, pp. 22–41, 2001.

[19] LEBLANC, M., TIBSHIRANI, R., "Adaptive principal surfaces", *Journal of the American Statistical Association*, v. 89, n. 425, pp. 53–64, 1994.

[20] KéGL, B., KRZYZAK, A., LINDER, T., *et al.*, "Learning and design of principal curves", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 22, n. 3, pp. 281–297, 2000.

[21] SANDILYA, S., KULKARNI, S., "Principal curves with bounded turn", *IEEE Transactions on Information Theory*, v. 48, n. 10, pp. 2789–2793, 2002.

[22] DELICADO, P., "Another look at principal curves and surfaces", *Journal of Multivariate Analysis*, v. 77, n. 1, pp. 84–116, 2001.

[23] DELICADO, P., HUERTA, M., "Principal curves of oriented points: theoretical and computational improvements", *Computational Statistics*, v. 18, n. 2, pp. 293–315, 2003.

[24] EINBECK, J., TUTZ, G., EVERS, L., *Local principal curves*, Report, SFB Discussion Report Number 320, 2003.

[25] HASTIE, T., TIBSHIRANI, R., *Generalized Additive Models*. Chapman and Hall, 1990.

[26] BISHOP, C. M., SVENSEN, M., *GTM: the generative topographic mapping*, Report, Technical Report NCRG/96/015, Aston University, 1997.

[27] KOHONEN, T., *Self-Organizing Maps*. Berlin, Heidelberg: Springer, 1995.

[28] MULIER, F., CHERKASSKY, V., "Self-organization as an iterative kernel smoothing process", *Neural Computation*, v. 7, n. 425, pp. 1165–1177, 1995.

[29] TAN, S., MAVROVOUNIOTIS, M. L., "Reducing data dimensionality through optimizing neural network inputs", *Journal of the American Institute of Chemical Engineering*, v. 41, n. 6, pp. 1471–1480, 1995.

[30] DONG, D., MCAVOY, T. J., "Nonlinear principal component analysis based on principal curves and neural networks", *Computers and Chemical Engineering*, v. 20, n. 1, pp. 65–78, 1996.

[31] TARPEY, T., LI, L., FLURY, B. D., "Principal points and self-consistent points of eliptical distributions", *Annals of Statistics*, v. 23(1), n. 6, pp. 103–112, 1995.

[32] STANFORD, D. C., RAFTERY, A. E., "Finding curvilinear features in spatial point patterns: principal curve clustering with noise", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 22, n. 6, pp. 601–609, 2000.

[33] KéGL, B., KRZYZAK, A., "Piecewise linear skeletonization using principal curves", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 22, n. 3, pp. 281–297, 2000.

[34] CHANG, K., GHOSH, J., "Principal curve classifier - a nonlinear approach to pattern classification", *Neural Networks Proceedings*, v. 1, pp. 695–700, 1998.

[35] VERBEEK, J. J., VLASSIS, N., KRÖSE, B., "A soft k-segments algorithm for principal curves", *Proceedings of International Conference on Artificial Neural Networks*, pp. 450–456, 2001.

[36] LOURENS, J., PREEZ, J. D., "Passive sonar ML estimator for ship propeller speed", *IEEE Journal of Oceanic Engineering*, v. 23, n. 4, pp. 448–453, 1998.

[37] RAJAGOPAL, R., SANKARANARAYANAN, K., RAO, P. R., "Target classification in a passive sonar - an expert system approach", *IEEE Intl. Conf. on Accoustic, Speech and Signal Processing*, v. 5, pp. 2911–2914, 1990.

[38] LI, Q., WANG, J., WEI, W., "An application of expert system in recognition of radiated noise of underwater targets", *IEEE Oceans - San Diego*, pp. 404–408, 1995.

[39] COTTLE, D., HAMILTON, D., "A neural sonar discrimination system", *IEEE Conference of Neural Networks for Ocean Engineering*, pp. 13–19, 1991.

[40] BARAN, R. H., COUGHLIN, J. P., "A neural network for target classification using passive sonar", *Association for Computing Machinery*, pp. 188–198, 1991.

[41] HAYKIN, S., *Neural Networks: A Comprehensive Foundation*. Prentice-Hall, 1999.

[42] HUANG, J., ZHAO, J., XIE, Y., "Source classification using pole method of AR method", *IEEE Intl. Conf. on Accoustic, Speech and Signal Processing*, v. 1, pp. 567–570, 1997.

[43] SOUZA-FILHO, J. B. D. O., SEIXAS, J. M., "Classificação de sinais de sonar passivo com filtros casados", *V Encontro de Tecnologia em Acústica Submarina*

(V-ETAS), pp. 189–194, Instituto de Pesquisas da Marinha, Rio de Janeiro, RJ, 2001.

[44] SILVA, M. R. P., SEIXAS, J. M., "Classificação de sinais de ruídos acústicos de navios empregando filtros casados", *I Encontro de Propagação e Acústica Submarina (I-EPAS)*, , Instituto de Pesquisas da Marinha, Rio de Janeiro, RJ, 2004.

[45] TREES, H. L. V., *Deteccion, Estimation and Modulation Theory, Part III*. John Wiley and Sons, 1971.

[46] RIBEIRO-FONSECA, J., CORREIA, L., "Identification of underwater acoustic noise", *IEEE Oceans*, pp. 597–602, 1994.

[47] RAJAGOPAL, R., KUMAR, K. A., RAO, P. R., "An integrated approach to passive target classification", *IEEE Intl. Conf. on Accoustic, Speech and Signal Processing*, v. 2, pp. II313–II316, 1994.

[48] NEVES-JR, I. P., *Classificação de alvos sonar passivo utilizando redes neurais*. M.Sc. dissertation, Universidade Federal do Rio de Janeiro, COPPE/UFRJ, 1995.

[49] SOARES-FILHO, W., SEIXAS, J. M. D., CALôBA, L. P., "Averaging spectra to improve the classification of the noise radiated by ships using neural networks", *VI Brazilian Symposium on Neural Networks, Rio de Janeiro*, v. 1, pp. 156–161, 2000.

[50] SOARES-FILHO, W., SEIXAS, J. M. D., CALôBA, L. P., "Principal component analysis for classifying passive sonar signals", *IEEE Intl. Conf. on Circuits and Systems, Sydney*, v. III, pp. 592–595, 2001.

[51] SOARES-FILHO, W., SEIXAS, J. M. D., CALôBA, L. P., "Enlarging neural class detection capacity in passive sonar systems", *IEEE Intl. Conf. on Circuits and Systems, Phoenix*, v. III, pp. 105–108, 2002.

[52] GHOSH, J., DEUSER, L., BECK, S., "A neural network based hybrid system for detection, characterization and classification of short-duration oceanic signals", *IEEE Journal of Oceanic Engineering*, v. 14, n. 4, pp. 351–363, 1991.

[53] CHEN, C., LEE, J., LIN, M., "Classification of underwater signals using wavelet transforms and neural networks", *Mathematical and Computer Modeling*, v. 27, n. 2, pp. 47–60, 1998.

[54] SEIXAS, J. M., DAMAZIO, D. O., DINIZ, P. S. R., *et al.*, "Wavelet transform as a preprocessing method for neural classification of passive sonar signals", *IEEE Intl. Conf. on Electronic Circuits and Systems, Saint Julians*, v. 1, pp. 83–86, 2001.

[55] GHOSH, J., DEUSER, L., "Classification of spatio-temporal patterns with applications to recognition of sonar sequences", 1995.

[56] AL., J. G. E., "Adaptive kernel classifiers for short-duration oceanic signals", *Proc. of IEEE Conf. on Neural Networks for Oceanic Eng.*, pp. 41–48, 1991.

[57] BROOMHEAD, D. S., LOWE, D., "Multivariable functional interpolation and adaptive networks", *Complex Systems*, v. 2, pp. 321–355, 1988.

[58] KOHONEN, T., BARNA, G., CHRISLEY, R., "Statistical pattern recognition with neural networks: Benchmarking studies", *IEEE Annual Intl. Conf. on Neural Networks*, , July, 1988.

[59] SHIN, Y., GHOSH, J., "The pi-sigma network: An efficient higher-order for pattern classification and function approximation", *Proceedings of the Intl. Joint Conf. on Neural Networks*, v. I, pp. 13–18, Seattle, 1991.

[60] REGAZZONI, C. S., TESEI, A., TACCONI, G., "A comparison between spectral and bispectral analysis for ship detection from Acoustical Time Series", *IEEE Intl. Conf.on Acoustics, Speech and Signal Processing*, pp. 289–292, Adelaide, 1994.

[61] LYONS, A. R., NEWTON, T. J., GODDARD, N. J., *et al.*, "Can passive sonar signals be classified on the basis of their higher-order statistics?", *Higher-Order Statistics in Signal Processing: Are They of Any Use?, IEE Colloquium on*, pp. 6/1–6/5, 1995.

[62] PFLUG, L. A., IOUP, G. E., IOUP, J. W., *et al.*, "Variability in higher order statistics of measured shallow-water shipping noise", *IEEE Signal Processing on Higher-Order Statistics*, pp. 400–404, Banff, 1997.

[63] GHOSH, J., BECK, S., CHU, C. C., "Evidence combination techniques for robust classification of short-duration oceanic signals", *SPIE Conference on Adaptive and learning Systems*, pp. 266–276, 1992.

[64] GHOSH, J., GANGISHETTI, N. V., CHAKRAVARTHY, S. V., "Robust classification of variable length sonar sequences", *Proceedings of SPIE - Applications of Artifitial Neural Networks IV*, pp. 96–107, 1993.

[65] GHOSH, J., DEUSER, L., "Neural representation of temporal patterns with applications to recognition of sonar sequences", *Plenum Publications*, pp. 227–250, 1995.

[66] HOWELL, B. P., WOOD, S., KOKSAL, S., "Passive sonar recognition and analysis using hybrid neural networks", College Park, MD, 1997.

[67] GHOSH, J., TUMER, K., BECK, S., *et al.*, "Integration of neural classifiers for passive sonar signals", 1995.

[68] COSTA, B. F. P., "Classificação sonar passiva utilizando redes neurais especialistas", 2004.

[69] THYAGARAJAN, K. S., NGUYEN, T., "An image processing approach to underwater acoustic signal classification", *IEEE Intl. Conf. on Systems, Man and Cybernetics*, v. 5, pp. 4198–4203, Orlando, 1997.

[70] VAIDYANATHAN, P. P., *Multirate Systems and Filter Banks*. Prentice Hall, 1993.

[71] DAUBECHIES, I., "Orthonormal bases of compactly supported wavelets", *Comm. on Pure and Appl. Math.*, v. 4, pp. 909–996, 1988.

[72] ALVEAR, C. A. S., SOARES-FILHO, W., SEIXAS, J. M., "Curva principal aplicada ao ruído irradiado por navios", *V Encontro de Tecnologia em Acústica*

Submarina (V-ETAS), Instituto de Pesquisas da Marinha, Rio de Janeiro, pp. 184–188, 2001.

[73] NIELSEN, R. O., *Sonar Signal Processing*. Artech House, 1991.

[74] SOUZA-FILHO, J. B. O., *Análise de componentes principais em sistemas sonar*. M.Sc. dissertation, Universidade Federal do Rio de Janeiro, COPPE/EE/UFRJ, 2002.

[75] FERNANDEZ, H. L., "Análise de dados de sonar passivo", 2003.

[76] BELOGAY, E., CABRELLI, C., MOLTER, V., *et al.*, "Calculating the Hausdorff distance between curves", *Information Processing Letters*, v. 64, pp. 17–22, 1997.

[77] SOUZA-FILHO, J. B. O., SEIXAS, J. M., "Agrupamento estatístico de dados de sonar passivo para o projeto de classificadores", *I Encontro de Tecnologia em Acústica Submarina (I-ETAS), Instituto de Pesquisas da Marinha, Rio de Janeiro*, , 2004.

[78] AMARAL, J. A., *Recognition of ship types from IR images using moment invariants and neural networks*. M.Sc. dissertation, Naval Postgraduate School - Monterey - CA - USA, 2001.

[79] NEVES, S. R., *Algoritmos para segmentação de imagens infravermelhas*. Phd thesis, Universidade Federal do rio de Janeiro, COPPE/EE/UFRJ, 2003.

[80] NERI, F., *Introduction to Electronic Defense Systems*. Artech House, 2001.

[81] SADJADI, F., CHUN, C., "Improved feature classification by means of polarimetric IR imaging sensor", *Geoscience and Remote Sensing Symposium*, v. 1, pp. 396–398, 1996.

[82] SADJADI, F., CHUN, C., "Passive polarimetric IR target classification", *IEEE Transactions on aerospace and electronic systems*, v. 37, pp. 740–750, 2001.

[83] HADDON, J., BOYCE, J. F., STRENS, M., "Autonomous segmentation and neural network texture classification of IR image sequences", 1996.

[84] KOTRLIK, M. J., REAL, E. C., "Adaptive threat detection and classification from dual band IR images", *Proceedings of the 2000 IEEE Sensor Array and Multichannel Signal Processing Workshop*, pp. 16–17, 2000.

[85] JAIN, A. K., *Fundamentals of Digital Image Processing*. Prentice Hall, 1989.

[86] BHARADWAJ, P., CARIN, L., "Infrared image classification using hidden Markov trees", *IEEE Transactions on aerospace and electronic systems*, v. 24, n. 10, pp. 1394–1398, 2002.

[87] NEVES, S. R., SILVA, E. A. B. D., MENDONçA, G. V., "Wavelet-watershed automatic infrared image segmentation method", *Electronic Letters*, v. 12, pp. 903–904, 2003.

Apêndice A

Códigos-fonte

Este anexo contém os códigos fontes desenvolvidos para a produção dos classificadores apresentados. Os códigos aqui descritos estão comentados, para facilitar sua compreensão.

Uma mesma rotina é empregada na classificação de espectros obtidos a partir do processamento de sinais de sonar e na classificação de informação de contorno obtida a partir das imagens de navios, as quais foram codificadas em Matlab6(R12) e em linguagem C. No primeiro caso, são usadas 8 classes e 557 dimensões, enquanto que, no segundo, são empregadas 5 classes e 300 ou 400 dimensões, de acordo com o tipo de ampeamento. Poucas alterações de código devem ser realizadas a fim de converter um tipo e classificador noutro, conforme pode ser verificado pela análise dos códigos fonte listados nas seções A.1 e A.2.

A.1 Classificador de navios baseado em sinais de sonar passivo

A.1.1 Rotina de classificação implementada em linguagem C

```
////////////////////////////////////////////////////////////////////////////
//
//      FILE: classif.c - versão 1
//      Sonar passivo
//
////////////////////////////////////////////////////////////////////////////

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

// Includes para os arquivos que contem as coordenadas dos vertices
// das curvas principais de cada classe. Sao arquivos de texto
```

```
// contendo,inicialmente, a descricao da matriz e, em seguida os
// valores das coordenadas entre virgulas. Exemplo:
//
// float classe1 [20] [557] = {
// 0.21351, 0.4578, ...
// };
// onde 20 e´o numero de seg. e 557 e´a dimensao dos dados.

#include "classe1.h"
#include "classe2.h"
#include "classe3.h"
#include "classe4.h"
#include "classe5.h"
#include "classe6.h"
#include "classe7.h"
#include "classe8.h"

////////////////////////////////////////////////////////////////////////////
// global variables
////////////////////////////////////////////////////////////////////////////

#define dimensao 557
#define nClasses 8
// quantidade de dados (espectros) do conjunto de teste = 12.
#define nDados   12

int nVertices[8]={10 33 22 6 29 17 9 12};  // Metodo AN3

////////////////////////////////////////////////////////////////////////////
// function prototypes
////////////////////////////////////////////////////////////////////////////

float sqdist( float * a, float * b)
{
  int i;
  float d2=0;

  for(i=0; i < dimensao; i++)
  {
    d2 = d2 + (a[i]-b[i])*(a[i]-b[i]);
  }

  return d2;
}

//-------------------------------------------------------
float segdist( float * v1, float * v2, float * x)
{
  float u[dimensao];
  float tvet[dimensao];
  float p[dimensao];
  float tamSegmento;
  float t;
  int k;
  float distqua;

  tamSegmento = sqrt(sqdist(v1,v2));
  t=0;

  for(k=0; k<dimensao; k++)
  {
    u[k] = (v2[k] - v1[k]) / tamSegmento;
    tvet[k] = x[k] - v1[k];
    t= t + (tvet[k] * u[k]);
  }

  if(t < 0) t=0;
  else if ( t > tamSegmento ) t = tamSegmento;

  for(k=0; k<dimensao; k++)
  {
    p[k] = v1[k] + (u[k] * t);
  }

  distqua = sqdist( x, p);

  return distqua;
}

//-------------------------------------------------------
float curvedist( float classe [] [dimensao], float * dado)
{
  float distSeg;
  float dist=1e10;
  int i;
  int segmento = -1;

  for (i=0; i<(nVertices[0]-1); i++)
  {
     distSeg = segdist(classe[i], classe[i+1], dado);
     if (distSeg < dist)
     {
        dist = distSeg;
        segmento = i;
     }
  }

  printf("distancia= %f segmento= %d \n", dist, segmento);
  return dist;
```

```
}

//-----------------------------------------------------
int main()
{

  float dist;
  int i,k,j;
  float * ptr[8];
  float menorDist;
  int   indClasse;

  ptr[0] = classe1[0];
  ptr[1] = classe2[0];
  ptr[2] = classe3[0];
  ptr[3] = classe4[0];
  ptr[4] = classe5[0];
  ptr[5] = classe6[0];
  ptr[6] = classe7[0];
  ptr[7] = classe8[0];


  for(k=0; k<12; k++)
  {
    menorDist = 1e10;
    for(i=0; i<8; i++)
    {
      dist = curvedist( (float(*)[dimensao]) ptr[i], dados[k]);
      printf("distancia para classe %d = %f \n", i+1, dist);
      if(dist < menorDist)
      {
        menorDist = dist;
        indClasse = i;
      }
    }


    printf("Ponto %d: obtida menor distancia %f para a classe %d \n",
            k+1, menorDist, indClasse+1);

  }
  return 0;

}
```

A.1.2 Rotina de classificação implementada em Matlab6

```
function [qtdes,estat]=classif_sonar(pathDados,pathCurvas,pathResultados)

% ======================================================
% Rotina de classificaçao.
% Variaveis de entrada:
% pathDados  = diretorio onde estao contidos os conjuntos de dados (projeto ou teste).
% pathCurvas = diretorio onde estao armazenados os arquivos vertices e edges que
%              caracterizam as curvas principais.
% pathResultados = indica o diretorio de armazenamento dos resultados de classificacao
%                  e histogramas de erro.
% Variaveis de saida:
% qtdes = matriz de confusao em valores absolutos.
% estat = matriz de confusao em porcentuais.
%
% Emprega as seguintes rotinas: map_to_arcl_mod.m, map_to_arcl.m e seg_dist.m
% (as duas ultimas foram desenvolvidas por J.J.Verbeek).
% Helena L. Fernandez
% ======================================================



disp(['dados:  ' pathDados]);
disp(['curvas: ' pathCurvas]);
disp(['resultados: ' pathResultados]);

disp('lendo segmentos...');

n_classes = 8;  % sonar

for i=1:n_classes
    arquivo = [pathCurvas 'edges' num2str(i)];
    eval(['edges' num2str(i) ' = dlmread(arquivo);']);
    arquivo = [pathCurvas 'vertices' num2str(i)];
    eval(['vertices' num2str(i) ' = dlmread(arquivo);']);
end

% ======================================================

result=zeros(n_classes,n_classes);
tamanhos=zeros(1,n_classes);

tic
for k=1:n_classes

  disp(['lendo dados de teste da classe #' num2str(k) ' ...']);
  arquivo = [pathDados 'teste' num2str(k) '.txt'];
  eval(['teste' num2str(k) ' = load(arquivo);']);
```

```
  dist=[];
  for i=1:n_classes
   disp(['verificando CP da classe #' num2str(i) ' ...']);
   d=[];
   eval(['[y,d]=map_to_arcl_mod(edges' num2str(i) ',vertices' num2str(i) ',teste' num2str(k) ');']);
   dist=[dist d];
  end

  menorIndice=[];
  [menorValor,menorIndice]=min(dist');

  [dummy, ndados]= size(menorIndice);
  tamanhos(1,k) = ndados;

  for i=1:ndados
    result(k,menorIndice(1,i))=result(k,menorIndice(1,i)) + 1;
  end

  % ====================================

  listaErros = [];
  posErro = [];
  curvaErro = [];
  for i=1:ndados
    if menorIndice(i) ~= k
        posErro = [posErro ; i];
        curvaErro = [curvaErro ; menorIndice(i)];
    end
  end

  listaErros = [posErro curvaErro];

  filename = [ pathResultados 'listaErros' num2str(k) '.csv' ];
  dlmwrite(filename, listaErros);

  % ====================================

  eval(['clear teste' num2str(k)]);
  clear y;
  clear d;
  disp(['-> tempo de processamento: ' num2str(toc/60) ' min']);
end

% ======================================================

qtdes = [result tamanhos'];

disp(' ');
disp('** Resultados sobre dados de teste **');
disp(' ');
disp('     Classificacao (qtdes) +  Total');
disp(qtdes);
estat = result;
for k=1:n_classes
  for i=1:n_classes
     estat(k,i)=estat(k,i)*100/tamanhos(1,k);
  end
end
disp(' ');
disp('   Classificacao (%)' );
disp(estat);


function [y,d]=map_to_arcl_mod(edges,vertices,x)

tam=size(x,1);
d=zeros(tam,1);
for i=1:tam
  [y1,d1]=map_to_arcl(edges,vertices,x(i,:));
  d(i,1)=d1;
end
y=[];


function [y,d]=map_to_arcl(edges,vertices,x)

% Map all datapoints to latent variable which is obtained by mapping point
% to closest point on path
% Path is indexed continuously in [0,l(P)] where l(P) is the length of the path

[n,d]=size(x);
segments=zeros(d,2,size(edges,1)-1);

e=edges; segment=1; lengths=zeros(size(segments,3)+1,1);
i=find(sum(e)==2);i=i(1);   %get an endpoint of path
j=find(e(i,:)>0);           % get neighbor

while segment <= size(segments,3)
  e(i,j)=0;e(j,i)=0;        % remove used edge
  segments(:,:,segment) = [vertices(:,i) vertices(:,j)];
  lengths(segment+1) = lengths(segment)+norm(vertices(:,i)-vertices(:,j));
  segment=segment+1;
  i=j;                      % find next segment
  j=find(e(i,:)>0);         % get neighbor
end

y=zeros(n,d+1); % labels in arc length (1-dim) + projected points (d-dim)
msqd=0;
```

```
dists = zeros(n,size(segments,3));
rest   = zeros(n,d+1,size(segments,3));
for i=1:size(segments,3)
  [d t p]=seg_dist(segments(:,1,i),segments(:,2,i),x');
  dists(:,i)=d;
  rest(:,:,i)=[t p];
end
[d,vr]=min(dists,[],2);
for i=1:n
y(i,:) = rest(i,:,vr(i));
y(i,1)=y(i,1)+lengths(vr(i));
end


function [d, t, p]=seg_dist(v1, v2, x)

% Compute the sq distance between x and the line segment from V1 to V2.
% d: sq distance
% p: projected point
% t: euclidean distance from projected point to v1

a = 0;
b = norm(v2-v1);

u = (v2-v1)/norm(b); % u = unit vector - segment is line v1 + tu with t in [0,b]

% get projection index on line (not segment!)
t = (x'-repmat(v1',size(x,2),1))*u;

% get projection index on segment
t = max(t,a);
t = min(t,b);

p = repmat(v1',size(x,2),1) + t*u';  % get projected points
d=(x'-p);
d=d.*d;
d=sum(d,2);




function d = sqdist(a,b)
% sqdist - computes pairwise squared Euclidean distances between points

% original version by Roland Bunschoten, 1999

aa = sum(a.*a,1); bb = sum(b.*b,1); ab = a'*b;
d = abs(repmat(aa',[1 size(bb,2)]) + repmat(bb,[size(aa,2) 1]) - 2*ab);
```

A.2 Classificador de navios baseado em imagens

A.2.1 Rotina de classificação implementada em linguagem C

```
////////////////////////////////////////////////////////////////////////////
//
//  FILE: classif.c - versão 1
//      Imagens
//
////////////////////////////////////////////////////////////////////////////

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

// Includes para os arquivos que contem as coordenadas dos vertices
// das curvas principais de cada classe. Sao arquivos de texto
// contendo,inicialmente, a descricao da matriz e, em seguida os
// valores das coordenadas entre virgulas. Exemplo:
//
// float classe1 [20] [300] = {
// 0.21351, 0.4578, ...
// };
// onde 20 e´o numero de seg. e 300 e´a dimensao dos dados.

#include "classe1.h"
#include "classe2.h"
#include "classe3.h"
#include "classe4.h"
#include "classe5.h"


////////////////////////////////////////////////////////////////////////////
// global variables
////////////////////////////////////////////////////////////////////////////

#define dimensao 300
```

```
#define nClasses 5
// quantidade de dados (espectros) do conjunto de teste = 12.
#define nDados   12

int nVertices[5]={26 22 18 17 19};  // Mapeamento do Tipo 1

////////////////////////////////////////////////////////////////////////////
// function prototypes
////////////////////////////////////////////////////////////////////////////

float sqdist( float * a, float * b)
{
  int i;
  float d2=0;

  for(i=0; i < dimensao; i++)
  {
    d2 = d2 + (a[i]-b[i])*(a[i]-b[i]);
  }

  return d2;
}

//-------------------------------------------------------
float segdist( float * v1, float * v2, float * x)
{
  float u[dimensao];
  float tvet[dimensao];
  float p[dimensao];
  float tamSegmento;
  float t;
  int k;
  float distqua;

  tamSegmento = sqrt(sqdist(v1,v2));
  t=0;

  for(k=0; k<dimensao; k++)
  {
    u[k] = (v2[k] - v1[k]) / tamSegmento;
    tvet[k] = x[k] - v1[k];
    t= t + (tvet[k] * u[k]);
  }

  if(t < 0) t=0;
  else if ( t > tamSegmento ) t = tamSegmento;

  for(k=0; k<dimensao; k++)
  {
    p[k] = v1[k] + (u[k] * t);
  }

  distqua = sqdist( x, p);

  return distqua;
}

//-------------------------------------------------------
float curvedist( float classe [] [dimensao], float * dado)
{
  float distSeg;
  float dist=1e10;
  int i;
  int segmento = -1;

  for (i=0; i<(nVertices[0]-1); i++)
  {
     distSeg = segdist(classe[i], classe[i+1], dado);
     if (distSeg < dist)
     {
        dist = distSeg;
        segmento = i;
     }
  }

  printf("distancia= %f segmento= %d \n", dist, segmento);
  return dist;
}

//-----------------------------------------------------
int main()
{

  float dist;
  int i,k,j;
  float * ptr[5];
  float menorDist;
  int   indClasse;

  ptr[0] = classe1[0];
  ptr[1] = classe2[0];
  ptr[2] = classe3[0];
  ptr[3] = classe4[0];
  ptr[4] = classe5[0];


  for(k=0; k<12; k++)
  {
    menorDist = 1e10;
```

```
    for(i=0; i<5; i++)
    {
      dist = curvedist( (float(*)[dimensao]) ptr[i], dados[k]);
      printf("distancia para classe %d = %f \n", i+1, dist);
      if(dist < menorDist)
      {
        menorDist = dist;
        indClasse = i;
      }
    }


    printf("Ponto %d: obtida menor distancia %f para a classe %d \n",
            k+1, menorDist, indClasse+1);

  }
  return 0;

}
```

A.2.2 Rotina de classificação implementada em Matlab6

```
function [qtdes,estat]=classif_vistas(pathDados,pathCurvas,pathResultados)

% ==================================================
% Rotina de classificaçao.
% Variaveis de entrada:
% pathDados  = diretorio onde estao contidos os conjuntos de dados (projeto ou teste).
% pathCurvas = diretorio onde estao armazenados os arquivos vertices e edges que
%              caracterizam as curvas principais.
% pathResultados = indica o diretorio de armazenamento dos resultados de classificacao
%                  e histogramas de erro.
% Variaveis de saida:
% qtdes = matriz de confusao em valores absolutos.
% estat = matriz de confusao em porcentuais.
%
% Emprega as seguintes rotinas: map_to_arcl_mod.m, map_to_arcl.m e seg_dist.m
% (as duas ultimas foram desenvolvidas por J.J.Verbeek).
% Helena L. Fernandez
% ==================================================



disp(['dados:  ' pathDados]);
disp(['curvas: ' pathCurvas]);
disp(['resultados: ' pathResultados]);

disp('lendo segmentos...');

n_classes = 5;  % vistas

for i=1:n_classes
    arquivo = [pathCurvas 'edges' num2str(i)];
    eval(['edges' num2str(i) ' = dlmread(arquivo);']);
    arquivo = [pathCurvas 'vertices' num2str(i)];
    eval(['vertices' num2str(i) ' = dlmread(arquivo);']);
end

% ==================================================

result=zeros(n_classes,n_classes);
tamanhos=zeros(1,n_classes);

tic
for k=1:n_classes

  disp(['lendo dados de teste da classe #' num2str(k) ' ...']);
  arquivo = [pathDados 'teste' num2str(k) '.txt'];
  eval(['teste' num2str(k) ' = load(arquivo);']);

  dist=[];
  for i=1:n_classes
   disp(['verificando CP da classe #' num2str(i) ' ...']);
   d=[];
   eval(['[y,d]=map_to_arcl_mod(edges' num2str(i) ',vertices' num2str(i) ',teste' num2str(k) ');']);
   dist=[dist d];
  end

  menorIndice=[];
  [menorValor,menorIndice]=min(dist');

  [dummy, ndados]= size(menorIndice);
  tamanhos(1,k) = ndados;

  for i=1:ndados
    result(k,menorIndice(1,i))=result(k,menorIndice(1,i)) + 1;
  end

  % ===================================

  listaErros = [];
  posErro = [];
  curvaErro = [];
  for i=1:ndados
```

```
    if menorIndice(i) ~= k
        posErro = [posErro ; i];
        curvaErro = [curvaErro ; menorIndice(i)];
    end
  end

  listaErros = [posErro curvaErro];

  filename = [ pathResultados 'listaErros' num2str(k) '.csv' ];
  dlmwrite(filename, listaErros);

  % ===================================

  eval(['clear teste' num2str(k)]);
  clear y;
  clear d;
  disp(['-> tempo de processamento: ' num2str(toc/60) ' min']);
end

% ====================================================

qtdes = [result tamanhos'];

disp(' ');
disp('** Resultados sobre dados de teste **');
disp(' ');
disp('     Classificacao (qtdes) +  Total');
disp(qtdes);
estat = result;
for k=1:n_classes
  for i=1:n_classes
     estat(k,i)=estat(k,i)*100/tamanhos(1,k);
  end
end
disp(' ');
disp('   Classificacao (%)' );
disp(estat);
```

A.2.3 Arquivos de modelos das classes de navios

```
function [verts,faces]=aircarrier();

verts=[122, -7,   0 %1
       122,   7,   0   %2
       116,   1,  -7   %3
       114,   0,  -12  %4
       116,  -1,  -7   %5
       77,  -10,   0   %6
       77,  -10,  -7   %7
      77,   -8, -12  %8
      68,  -10,  -7  %9
      68,  -21,   0    %10
      -23, -21,   0    %11
      -23, -10,  -7    %12
      -62, -10,   0    %13
      -62, -10,  -7    %14
      -60,  -8, -12   %15
      -62,  10,   0    %16
      -62,  10,  -7    %17
      -60,   8, -12   %18
      -36,  12,   0    %19
      -36,  10,  -7    %20
      -28,  21,   0    %21
      -28,  10,  -7    %22
      38,   21,   0    %23
      38,   10,  -7    %24
      65,   24,   0    %25
      65,   10,  -7    %26
      77,   10,   0    %27
      77,   10,  -7    %28
      77,    8, -12    %29
      -7,  -19,   0    %30
      -7,  -19,   9    %31
      -7,  -15,   9    %32
      -7,  -15,   0    %33
      7,   -19,   0    %34
      7,   -19,   9    %35
      7,   -12,   9    %36
      7,   -12,   0    %37
      -1,  -17,   9    %38
      -1,  -17,  20    %39
      -1,  -15,  20    %40
      -1,  -15,   9    %41
      1,   -17,   9    %42
      1,   -17,  20    %43
      1,   -15,   9    %44
      1,   -15,  20];%45

   %number =       32 12

   plane1  =   [1,2,3,4,5,5,5,5,5,5,5,5];
   plane2  =   [1,5,6,6,6,6,6,6,6,6,6,6];
   plane3  =   [5,6,7,7,7,7,7,7,7,7,7,7];
   plane4  =   [4,5,7,8,8,8,8,8,8,8,8,8];
   plane5  =   [6,9,7,7,7,7,7,7,7,7,7,7];
```

```
    plane6  =   [6,10,9,9,9,9,9,9,9,9,9];
    plane7  =   [9,10,11,12,12,12,12,12,12,12,12];
    plane8  =   [11,12,13,13,13,13,13,13,13,13,13];
    plane9  =   [12,13,14,14,14,14,14,14,14,14,14];
    plane10 =   [7,9,12,14,15,8,8,8,8,8,8];
    plane11 =   [13,16,17,14,14,14,14,14,14,14,14];
    plane12 =   [14,17,18,15,15,15,15,15,15,15,15];
    plane13 =   [16,19,20,17,17,17,17,17,17,17,17];
    plane14 =   [19,21,22,20,20,20,20,20,20,20,20];
    plane15 =   [21,23,24,22,22,22,22,22,22,22,22];
    plane16 =   [23,25,26,24,24,24,24,24,24,24,24];
    plane17 =   [25,27,26,26,26,26,26,26,26,26,26];
    plane18 =   [26,27,28,28,28,28,28,28,28,28,28];
    plane19 =   [27,3,28,28,28,28,28,28,28,28,28];
    plane20 =   [27,2,3,3,3,3,3,3,3,3,3];
    plane21 =   [28,3,4,29,29,29,29,29,29,29,29];
    plane22 =   [17,20,22,24,26,28,29,18,18,18,18];
    flightdeck  =   [1,6,10,11,13,16,19,21,23,25,27,2];
    bottom  =   [4,8,15,18,29,29,29,29,29,29,29];
    tower1  =   [30,31,32,33,33,33,33,33,33,33,33];
    tower2  =   [30,31,35,34,34,34,34,34,34,34,34];
    tower3   =  [34,35,36,37,37,37,37,37,37,37,37];
    tower4  =   [32,33,37,36,36,36,36,36,36,36,36];
    tower5  =   [31,32,36,35,35,35,35,35,35,35,35];
    mast1   =   [38,39,40,41,41,41,41,41,41,41,41];
    mast2   =   [38,39,43,42,42,42,42,42,42,42,42];
    mast3   =   [42,44,45,43,43,43,43,43,43,43,43];
    mast4   =   [41,44,45,40,40,40,40,40,40,40,40];
    mast5   =   [39,43,45,40,40,40,40,40,40,40,40];
   %faces

   faces=[  plane1
    plane2
    plane3
    plane4
    plane5
    plane6
    plane7
    plane8
   plane9
   plane10
    plane11
    plane12
    plane13
    plane14
    plane15
    plane16
    plane17
    plane18
    plane19
    plane20
    plane21
    plane22
    flightdeck
    bottom
    tower1
    tower2
    tower3
    tower4
    tower5
    mast1
    mast2
    mast3
    mast4
    mast5 ];


function [verts,faces]=destroyer();

verts=[140, 0,   5  %1
        90,   -16, 3    %2
        90,   -16,  -10 %3
        125,   0,  -10  %4
        50,  -20,  0    %5
        50,  -20,   -10 %6
        44,  -20,  0    %7
       44,   -20, 5  %8
       38,  -20,  5  %9
       29,  -20,   5    %10
       8, -20,   5  %11
       8, -20,  0   %12
       -8, -20,   0 %13
       -8, -20,  -10    %14
       -8,  -20, 5   %15
       -55,  -14,   5   %16
       -55,  -14,  -3   %17
       -55,   -13, -10   %18
       -85,  -10,   -3  %19
       -84,  -9,  -10   %20
       -85,  10,   -3   %21
       -84,  9,  -10    %22
       -55,   14,   -3  %23
       -55,   13,  -10  %24
       -55,   14,   5   %25
       -8,   20,  5 %26
       -8,   20,   0    %27
       -8,   20,  -10   %28
       8,    20, 0  %29
       8,  20,   5  %30
```

```
  29,  20,   5 %31
  38,  20,   5 %32
  44,  20,   5 %33
  44,   20,    0    %34
  50,   20,    0    %35
  50,   20,   -10  %36
  90,   16,    3    %37
  90,  15,   -10   %38
  50,  -11,  0 %39
  50,  11,  0  %40
  50,  -6,   5 %41
  44,   -6,    5    %42
  44,   -11,  5    %43
  38,   -20,   5   %44
 44,   -20,  5%45
 50,   -11,  5%46
 50,     6,  5%47
 44,     6,  5%48
 44,    11,  5%49
 38,    20,  5%50
 44,    20,  5%51
 50,    11,  5%52
 50,    -6, 14%53
 44,    -6, 14%54
 44,     6, 14%55
 50,     6, 14%56
 38,   -19, 20%57
 44,   -11, 20%58
 44,    11, 20%59
 38,    19, 20%60
 29,    19, 20%61
 25,    11, 20%62
 25,   -11, 20%63
 29,   -19, 20%64
 0,    -10,  5%65
 20,   -10,  5%66
 20,    10,  5%67
 0,     10,  5%68
 24,    11,  5%69
 24,   -11,  5%70
 6,     -4, 20%71
 14,    -4, 20%72
 14,     4, 20%73
 6,      4, 20%74
 0,    -10,  0%75
 0,     10,  0%76
 -32,  -10,  5%77
 -8,   -10,  5%78
 -8,    10,  5%79
 -32,   10,  5%80
 -22,   -4, 20%81
 -14,   -4, 20%82
 -14,    4, 20%83
 -22,    4, 20%84
 29,    -2, 20%85
 33,    -2, 20%86
 33,     2, 20%87
 29,     2, 20%88
 25,    -2, 50%89
 29,    -2, 50%90
 29,     2, 50%91
 25,     2, 50];%92


%number =       32 12

plane1  =   [1,2,3,4,4,4,4,4,4,4,4];
plane2  =   [2,5,6,3,3,3,3,3,3,3,3];
plane3  =   [5,7,8,9,10,11,12,13,14,6,6];
plane4  =   [13,15,16,17,18,14,14,14,14,14,14];
plane5  =   [17,19,20,18,18,18,18,18,18,18,18];
plane6  =   [19,21,22,20,20,20,20,20,20,20,20];
plane7  =   [21,23,24,22,22,22,22,22,22,22,22];
plane8  =   [23,25,26,27,28,24,24,24,24,24,24];
plane9  =   [27,29,30,31,32,33,34,35,36,28,28];
plane10 =   [35,37,38,36,36,36,36,36,36,36,36];
plane11 =   [37,1,4,38,38,38,38,38,38,38,38];
botton  =   [4,3,6,14,18,20,22,24,28,36,38];
deck1   =   [1,2,37,37,37,37,37,37,37,37,37];
deck2   =   [2,5,35,37,37,37,37,37,37,37,37];
deck3   =   [5,7,39,39,39,39,39,39,39,39,39];
deck4   =   [34,35,40,40,40,40,40,40,40,40,40];
deck5   =   [46,41,42,43,9,8,8,8,8,8,8];
deck6   =   [47,48,49,32,51,52,52,52,52,52,52];
deck7   =   [53,54,55,56,56,56,56,56,56,56,56];
deck8   =   [57,58,59,60,61,62,63,64,64,64,64];
deck9   =   [10,11,65,66,67,68,30,31,69,70,70];
deck10  =   [13,12,75,76,29,27,27,27,27,27,27];
deck11  =   [16,15,78,77,80,79,26,25,25,25,25];
deck12  =   [19,17,23,21,21,21,21,21,21,21,21];
lateral1    =   [8,46,39,7,7,7,7,7,7,7,7];
lateral2    =   [39,46,41,53,56,47,52,40,40,40,40];
lateral3    =   [40,52,33,34,34,34,34,34,34,34,34];
lateral4    =   [54,53,41,42,42,42,42,42,42,42,42];
lateral5    =   [55,56,47,48,48,48,48,48,48,48,48];
lateral6    =   [58,59,49,48,55,54,42,43,43,43,43];
lateral7    =   [57,58,43,9,9,9,9,9,9,9,9];
lateral8    =   [60,59,49,32,32,32,32,32,32,32,32];
```

```
 lateral9    =   [64,57,9,10,10,10,10,10,10,10,10];
lateral10=   [61,60,32,31,31,31,31,31,31,31,31];
lateral11=   [63,64,10,70,70,70,70,70,70,70,70];
 lateral12=  [62,61,31,69,69,69,69,69,69,69,69];
 lateral13=  [62,63,70,69,69,69,69,69,69,69,69];
 lateral14=  [65,11,12,75,75,75,75,75,75,75,75];
 lateral15=  [68,30,29,76,76,76,76,76,76,76,76];
 lateral16=  [65,75,76,68,68,68,68,68,68,68,68];
 lateral17=  [15,13,27,26,26,26,26,26,26,26,26];
lateral18=   [16,17,23,25,25,25,25,25,25,25,25];
stack1   =  [71,72,66,65,65,65,65,65,65,65,65];
stack2   =  [74,73,67,68,68,68,68,68,68,68,68];
stack3   =  [65,71,74,68,68,68,68,68,68,68,68];
stack4   =  [66,72,73,67,67,67,67,67,67,67,67];
stack5   =  [71,72,73,74,74,74,74,74,74,74,74];
stack6   =  [77,81,82,78,78,78,78,78,78,78,78];
stack7   =  [80,84,83,79,79,79,79,79,79,79,79];
stack8   =  [77,81,84,80,80,80,80,80,80,80,80];
stack9   =  [78,82,83,79,79,79,79,79,79,79,79];
stack10  =  [81,82,83,84,84,84,84,84,84,84,84];
mast1    =  [85,86,90,89,89,89,89,89,89,89,89];
mast2    =  [86,87,91,90,90,90,90,90,90,90,90];
mast3    =  [87,88,92,91,91,91,91,91,91,91,91];
mast4    =  [88,85,89,92,92,92,92,92,92,92,92];
mast5    =  [89,90,91,92,92,92,92,92,92,92,92];

%faces

faces=[plane1
   plane2
   plane3
   plane4
   plane5
   plane6
   plane7
   plane8
   plane9
   plane10
   plane11
   botton
   deck1
   deck2
   deck3
   deck4
   deck5
   deck6
   deck7
   deck8
   deck9
   deck10
   deck11
   deck12
   lateral1
   lateral2
   lateral3
   lateral4
   lateral5
   lateral6
   lateral7
   lateral8
   lateral9
   lateral10
   lateral11
   lateral12
   lateral13
   lateral14
   lateral15
   lateral16
   lateral17
   lateral18
   stack1
   stack2
   stack3
   stack4
   stack5
   stack6
   stack7
   stack8
   stack9
   stack10
   mast1
   mast2
   mast3
   mast4
   mast5];


function [verts,faces]=frigate();

verts=[128, 0,   16 %1
       96,   -13, 14   %2
       96,   -7,  0    %3
       112,   0,  0    %4
       92,  -15,  12   %5
       92,  -9,   0    %6
       60,  -20,  10   %7
      60,   -18, 0  %8
      58,  -20,  10  %9
      48,  -20,  10    %10
```

```
 -26, -20,  10    %11
 -60, -20,  10    %12
 -60, -18,   0    %13
 -102,-15,  10    %14
 -98,  -13,  0   %15
 -102,  15, 10    %16
 -98,  13,  0 %17
 -60,   20, 10   %18
 -60,  18,   0    %19
 -26,  20,  10    %20
 48,  20,   10    %21
 58,  20,   10    %22
 60,  20,   10    %23
 60,18,0  %24
 92,15,12 %25
 92,9,0   %26
 96,13,14 %27
 96,7,0   %28
 60,-13,10    %29
 60,13,10 %30
 -12,-12,10   %31
 8,-12,10 %32
 8,-16,10 %33
 -12,12,10    %34
 8,12,10  %35
 8,16,10  %36
 58,-20,20    %37
 48,-20,20    %38
 48,-16,20    %39
 8,-16,20 %40
 8,-12,20 %41
 -12,-12,20   %42
 -26,-20,20   %43
 -60,-20,20   %44
-60,20,20   %45
-26,20,20   %46
-12,12,20   %47
8,12,20     %48
8,16,20     %49
48,16,20    %50
48,20,20    %51
58,20,20    %52
60,13,20    %53
60,-13,20   %54
57,-13,25   %55
43,-13,25   %56
43,13,25    %57
57,13,25    %58
60,7,25     %59
60,-7,25    %60
49,-20,20   %61
49,-20,12   %62
51,-20,12   %63
51,-20,20   %64
52,-20,20   %65
52,-20,12   %66
56,-20,22   %67
58,-20,22   %68
49,20,20    %69
49,20,12    %70
51,20,12    %71
51,20,20    %72
52,20,20    %73
52,20,12    %74
56,20,22    %75
58,20,22    %76
60,-13,22   %77
60,-7,22    %78
60,7,22     %79
60,13,22    %80
60,-7,20    %81
57,-13,20   %82
60,7,20     %83
57,13,20    %84
43,-13,20   %85
43,13,20    %86
59,-16,20   %87
59,-16,10   %88
59,16,20    %89
59,16,10    %90
-39,-5,20   %91
-37,-4,25   %92
-37,4,25    %93
-39,5,20    %94
-29,-4,25   %95
-28,-5,20   %96
-29,4,25    %97
-28,5,20    %98
-16,-3,24   %99
-10,-3,24    %100
-10,-3,20    %101
-16,-3,20   %102
-16,3,24    %103
-10,3,24     %104
-10,3,20     %105
-16,3,20    %106
18,-1,48    %107
20,-1,48    %108
20,1,48     %109
```

```
   18,1,48     %110
   18,-1,20    %111
   20,-1,20    %112
   20,1,20     %113
   18,1,20     %114
   36,-2,42    %115
   39,-2,42    %116
   39,2,42     %117
   36,2,42     %118
   36,-2,20    %119
   39,-2,20    %120
   39,2,20     %121
   36,2,20     %122
   48,-2,33    %123
   52,-2,33    %124
   52,2,33     %125
   48,2,33     %126
   48,-2,25    %127
   52,-2,25    %128
   52,2,25     %129
   48,2,25];     %130


 %number =       66 planes

 plane1  =   [1,2,3,4,4,4,4,4,4,4,4,4,4,4,4,4,4];
 plane2  =   [2,5,6,3,3,3,3,3,3,3,3,3,3,3,3,3,3];
 plane3  =   [5,7,8,6,6,6,6,6,6,6,6,6,6,6,6,6,6];
 plane4  =   [7,9,10,11,12,13,8,8,8,8,8,8,8,8,8,8,8];
 plane5  =   [12,14,15,13,13,13,13,13,13,13,13,13,13,13,13,13,13];
 plane6  =   [14,16,17,15,15,15,15,15,15,15,15,15,15,15,15,15,15];
 plane7  =   [16,18,19,17,17,17,17,17,17,17,17,17,17,17,17,17,17];
 plane8  =   [18,20,21,22,23,24,19,19,19,19,19,19,19,19,19,19,19];
 plane9  =   [23,25,26,24,24,24,24,24,24,24,24,24,24,24,24,24,24];
 plane10 =   [25,27,28,26,26,26,26,26,26,26,26,26,26,26,26,26,26];
 plane11 =   [27,1,4,28,28,28,28,28,28,28,28,28,28,28,28,28,28];
 botton  =   [4,3,6,8,13,15,17,19,24,26,28,28,28,28,28,28,28];
 deck1   =   [1,2,27,27,27,27,27,27,27,27,27,27,27,27,27,27,27];
 deck2   =   [2,5,25,27,27,27,27,27,27,27,27,27,27,27,27,27,27];
 deck3   =   [5,7,9,29,30,22,23,25,25,25,25,25,25,25,25,25,25];
 deck4   =   [9,10,11,31,32,33,88,88,88,88,88,88,88,88,88,88,88];
 deck5   =   [22,21,20,34,35,36,90,90,90,90,90,90,90,90,90,90,90];
 deck6   =   [37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54];
 deck7   =   [12,14,16,18,18,18,18,18,18,18,18,18,18,18,18,18,18];
 deck8   =   [55,56,57,58,59,60,60,60,60,60,60,60,60,60,60,60,60];
 lateral1    =   [9,10,38,61,62,63,64,65,66,67,68,68,68,68,68,68,68];
 lateral2    =   [22,21,51,69,70,71,72,73,74,75,76,76,76,76,76,76,76];
 lateral3    =   [77,78,60,59,79,80,30,29,29,29,29,29,29,29,29,29,29];
 lateral4    =   [68,77,29,9,9,9,9,9,9,9,9,9,9,9,9,9,9];
 lateral5    =   [76,80,30,22,22,22,22,22,22,22,22,22,22,22,22,22,22];
 lateral6    =   [55,60,81,82,82,82,82,82,82,82,82,82,82,82,82,82,82];
 lateral7    =   [58,59,83,84,84,84,84,84,84,84,84,84,84,84,84,84,84];
 lateral8    =   [56,55,82,85,85,85,85,85,85,85,85,85,85,85,85,85,85];
 lateral9    =   [57,58,84,86,86,86,86,86,86,86,86,86,86,86,86,86,86];
lateral10=   [56,57,86,85,85,85,85,85,85,85,85,85,85,85,85,85,85];
lateral11=   [87,88,33,40,40,40,40,40,40,40,40,40,40,40,40,40,40];
 lateral12=  [89,90,36,49,49,49,49,49,49,49,49,49,49,49,49,49,49];
 lateral13=  [40,33,32,41,41,41,41,41,41,41,41,41,41,41,41,41,41];
 lateral14=  [49,36,35,48,48,48,48,48,48,48,48,48,48,48,48,48,48];
 lateral15=  [31,42,41,32,32,32,32,32,32,32,32,32,32,32,32,32,32];
 lateral16=  [34,47,48,35,35,35,35,35,35,35,35,35,35,35,35,35,35];
 lateral17=  [43,11,31,42,42,42,42,42,42,42,42,42,42,42,42,42,42];
lateral18=   [46,20,34,47,47,47,47,47,47,47,47,47,47,47,47,47,47];
lateral19=   [44,43,11,12,12,12,12,12,12,12,12,12,12,12,12,12,12];
lateral20=   [45,46,20,18,18,18,18,18,18,18,18,18,18,18,18,18,18];
lateral21=   [44,12,18,45,45,45,45,45,45,45,45,45,45,45,45,45,45];
stack1   =  [91,92,93,94,94,94,94,94,94,94,94,94,94,94,94,94,94];
stack2   =  [91,92,95,96,96,96,96,96,96,96,96,96,96,96,96,96,96];
stack3   =  [96,95,97,98,98,98,98,98,98,98,98,98,98,98,98,98,98];
stack4   =  [94,93,97,98,98,98,98,98,98,98,98,98,98,98,98,98,98];
stack5   =  [92,95,97,93,93,93,93,93,93,93,93,93,93,93,93,93,93];
turret1  =  [99,100,101,102,102,102,102,102,102,102,102,102,102,102,102,102,102];
turret2  =  [100,104,105,101,101,101,101,101,101,101,101,101,101,101,101,101,101];
turret3  =  [103,104,105,106,106,106,106,106,106,106,106,106,106,106,106,106,106];
turret4  =  [99,102,106,103,103,103,103,103,103,103,103,103,103,103,103,103,103];
turret5  =  [99,100,104,103,103,103,103,103,103,103,103,103,103,103,103,103,103];
mast1    =  [107,108,112,111,111,111,111,111,111,111,111,111,111,111,111,111,111];
mast2    =  [108,109,113,112,112,112,112,112,112,112,112,112,112,112,112,112,112];
mast3    =  [109,110,114,113,113,113,113,113,113,113,113,113,113,113,113,113,113];
mast4    =  [110,107,111,114,114,114,114,114,114,114,114,114,114,114,114,114,114];
mast5    =  [107,108,109,110,110,110,110,110,110,110,110,110,110,110,110,110,110];
surface1 =  [115,116,120,119,119,119,119,119,119,119,119,119,119,119,119,119,119];
surface2 =  [116,117,121,120,120,120,120,120,120,120,120,120,120,120,120,120,120];
surface3 =  [117,118,122,121,121,121,121,121,121,121,121,121,121,121,121,121,121];
surface4 =  [118,115,119,122,122,122,122,122,122,122,122,122,122,122,122,122,122];
surface5 =  [115,116,117,118,118,118,118,118,118,118,118,118,118,118,118,118,118];
radar1   =  [123,124,128,127,127,127,127,127,127,127,127,127,127,127,127,127,127];
radar2   =  [124,125,129,128,128,128,128,128,128,128,128,128,128,128,128,128,128];
radar3   =  [125,126,130,129,129,129,129,129,129,129,129,129,129,129,129,129,129];
radar4   =  [126,123,127,130,130,130,130,130,130,130,130,130,130,130,130,130,130];
radar5   =  [123,124,125,126,126,126,126,126,126,126,126,126,126,126,126,126,126];

%faces  66 planes

faces=[plane1
   plane2
   plane3
   plane4
   plane5
```

```
      plane6
      plane7
      plane8
      plane9
      plane10
      plane11
      botton
      deck1
      deck2
      deck3
      deck4
      deck5
      deck6
      deck7
      deck8
      lateral1
      lateral2
      lateral3
      lateral4
      lateral5
      lateral6
      lateral7
      lateral8
      lateral9
      lateral10
      lateral11
      lateral12
      lateral13
      lateral14
      lateral15
      lateral16
      lateral17
      lateral18
      lateral19
      lateral20
      lateral21
      stack1
      stack2
      stack3
      stack4
      stack5
      turret1
      turret2
      turret3
      turret4
      turret5
      mast1
      mast2
      mast3
      mast4
      mast5
      surface1
      surface2
      surface3
      surface4
      surface5
      radar1
      radar2
      radar3
      radar4
      radar5];


function [verts,faces]=merchant();

verts=[115,-11,0    %1
   115,-15,10       %2
   118.9,-14.5,10.8 %3
   117.8,-10.6,0    %4
   122.5,-13,11.5   %5
   120.5,-9.5,0     %6
   125.6,-10.6,12   %7
   122.8,-7.8,0     %8
   128,-7.5,12.5    %9
   124.5,-5.5,0     %10
   129.5,-3.9,13    %11
   125.6,-2.85,0    %12
   130,0,13         %13
   126,0,0          %14
   129.5,3.9,13     %15
   125.6,2.85,0     %16
   128,7.5,12.5     %17
   124.5,5.5,0      %18
   125.6,10.6,12    %19
   122.8,7.8,0      %20
   122.5,13,11.5    %21
   120.5,9.5,0      %22
   118.9,14.5,10.8  %23
   117.8,10.6,0     %24
   115,15,10        %25
   115,11,0         %26
   96,-15,10        %27
   93,-15,7         %28
   -130,-15,7       %29
   -128,-11,0       %30
   -128,11,0        %31
   -130,15,7        %32
   93,15,7          %33
   96,15,10         %34
```

```
-80,-15,23        %35
-80,15,23         %36
-84,15,23         %37
-84,11,23         %38
-91,11,23         %39
-91,-11,23        %40
-84,-11,23        %41
-84,-15,23        %42
-91,-11,17        %43
-91,11,17         %44
-110,11,17        %45
-110,-11,17       %46
-110,11,7         %47
-110,-11,7         %48
-80,-15,21        %49
-80,-11,19        %50
-80,-11,7         %51
-80,11,7          %52
-80,11,19         %53
-80,15,21         %54
-84,-15,21        %55
-84,-11,19        %56
-91,-11,7         %57
-84,15,21         %58
-84,11,19         %59
-91,11,7          %60
-99,-3,17         %61
-99,3,17          %62
-104,3,17         %63
-104,-3,17        %64
-99,3,27          %65
-99,-3,27         %66
-104,3,27         %67
-104,-3,27        %68
2,-11,7           %69
2,-9,7            %70
0,-9,7            %71
0,-11,7           %72
2,-11,20          %73
2,-9,20           %74
0,-11,20          %75
0,-9,20           %76
2,9,7             %77
2,11,7            %78
0,11,7            %79
0,9,7             %80
2,9,20            %81
2,11,20           %82
0,11,20           %83
0,9,20            %84
111,-1,10         %85
111,1,10          %86
109,1,10          %87
109,-1,10         %88
111,-1,26         %89
111,1,26          %90
109,1,26          %91
109,-1,26         %92
-89,-1,23         %93
-89,1,23          %94
-91,1,23          %95
-91,-1,23         %96
-89,-1,30         %97
-89,1,30          %98
-91,1,30          %99
-91,-1,30];       %100

 %planes

proa1 = [1,2,3,4,4,4,4,4,4,4,4,4,4,4,4];
proa2 = [3,5,6,4,4,4,4,4,4,4,4,4,4,4,4];
proa3 = [5,7,8,6,6,6,6,6,6,6,6,6,6,6,6];
proa4 = [7,9,10,8,8,8,8,8,8,8,8,8,8,8,8];
proa5 = [9,11,12,10,10,10,10,10,10,10,10,10,10,10,10];
proa6 = [11,13,14,12,12,12,12,12,12,12,12,12,12,12,12];
proa7 = [13,15,16,14,14,14,14,14,14,14,14,14,14,14,14];
proa8 = [15,17,18,16,16,16,16,16,16,16,16,16,16,16,16];
proa9 = [17,19,20,18,18,18,18,18,18,18,18,18,18,18,18];
proa10= [19,21,22,20,20,20,20,20,20,20,20,20,20,20,20];
proa11= [21,23,24,22,22,22,22,22,22,22,22,22,22,22,22];
proa12= [23,25,26,24,24,24,24,24,24,24,24,24,24,24,24];

plane1 = [1,2,27,28,29,30,30,30,30,30,30,30,30,30,30];
plane2 = [29,30,31,32,32,32,32,32,32,32,32,32,32,32,32];
plane3 = [26,25,34,33,32,31,31,31,31,31,31,31,31,31,31];

deck1 = [2,3,5,7,9,11,13,15,17,19,21,23,25,25,25];
deck2 = [2,25,34,27,27,27,27,27,27,27,27,27,27,27,27];
deck3 = [27,34,33,28,28,28,28,28,28,28,28,28,28,28,28];
deck4 = [28,33,32,29,29,29,29,29,29,29,29,29,29,29,29];
deck5 = [35,36,37,38,39,40,41,42,42,42,42,42,42,42,42];
deck6 = [40,39,44,43,43,43,43,43,43,43,43,43,43,43,43];
deck7 = [43,44,45,46,46,46,46,46,46,46,46,46,46,46,46];
deck8 = [46,45,47,48,48,48,48,48,48,48,48,48,48,48,48];

botton = [1,4,6,8,10,12,14,16,18,20,22,24,26,31,30];

front = [35,49,50,51,52,53,54,36,36,36,36,36,36,36,36];
lateral1 = [51,50,56,41,40,43,57,57,57,57,57,57,57,57,57];
```

```
lateral2 = [52,53,59,38,39,44,60,60,60,60,60,60,60,60];
lateral3 = [43,46,48,57,57,57,57,57,57,57,57,57,57,57];
lateral4 = [44,45,47,60,60,60,60,60,60,60,60,60,60,60];
blocoright1 = [35,49,55,42,42,42,42,42,42,42,42,42,42,42];
blocoright2 = [41,42,55,56,56,56,56,56,56,56,56,56,56,56];
blocoleft1 = [36,37,58,54,54,54,54,54,54,54,54,54,54,54];
blocoleft2 = [37,38,59,58,54,54,54,54,54,54,54,54,54,54];

stack1 = [64,61,66,68,68,68,68,68,68,68,68,68,68,68];
stack2 = [61,62,65,66,66,66,66,66,66,66,66,66,66,66];
stack3 = [62,65,67,63,63,63,63,63,63,63,63,63,63,63];
stack4 = [67,68,64,63,63,63,63,63,63,63,63,63,63,63];
stacktop = [66,65,67,68,68,68,68,68,68,68,68,68,68,68];

middleright1 = [72,69,73,75,75,75,75,75,75,75,75,75,75,75];
middleright2 = [69,70,74,73,73,73,73,73,73,73,73,73,73,73];
middleright3 = [70,74,76,71,71,71,71,71,71,71,71,71,71,71];
middleright4 = [71,76,75,72,72,72,72,72,72,72,72,72,72,72];
middlerighttop = [73,74,75,76,76,76,76,76,76,76,76,76,76,76];

middleleft1 = [80,77,81,84,84,84,84,84,84,84,84,84,84,84];
middleleft2 = [77,78,82,81,81,81,81,81,81,81,81,81,81,81];
middleleft3 = [78,79,83,82,82,82,82,82,82,82,82,82,82,82];
middleleft4 = [79,80,84,83,83,83,83,83,83,83,83,83,83,83];
middlelefttop = [81,82,83,84,84,84,84,84,84,84,84,84,84,84];

mastproa1 = [88,85,89,92,92,92,92,92,92,92,92,92,92,92];
mastproa2 = [85,86,90,89,89,89,89,89,89,89,89,89,89,89];
mastproa3 = [86,87,91,90,90,90,90,90,90,90,90,90,90,90];
mastproa4 = [87,88,92,91,91,91,91,91,91,91,91,91,91,91];
mastproatop = [89,90,91,92,92,92,92,92,92,92,92,92,92,92];

mastpopa1 = [96,93,97,100,100,100,100,100,100,100,100,100,100,100];
mastpopa2 = [93,94,98,97,97,97,97,97,97,97,97,97,97,97];
mastpopa3 = [94,98,99,95,95,95,95,95,95,95,95,95,95,95];
mastpopa4 = [95,99,100,96,96,96,96,96,96,96,96,96,96,96];
mastpopatop = [97,98,99,100,100,100,100,100,100,100,100,100,100,100];

%faces

faces=[proa1
   proa2
   proa3
   proa4
   proa5
   proa6
   proa7
   proa8
   proa9
   proa10
   proa11
   proa12
   plane1
   plane2
   plane3
   deck1
   deck2
   deck3
   deck4
   deck5
   deck6
   deck7
   deck8
   botton
   front
   lateral1
   lateral2
   lateral3
   lateral4
   blocoright1
   blocoright2
   blocoleft1
   blocoleft2
   stack1
   stack2
   stack3
   stack4
   stacktop
   middleright1
   middleright2
   middleright3
   middleright4
   middlerighttop
   middleleft1
   middleleft2
   middleleft3
   middleleft4
   middlelefttop
   mastproa1
   mastproa2
   mastproa3
   mastproa4
   mastproatop
   mastpopa1
   mastpopa2
   mastpopa3
   mastpopa4
   mastpopatop];
```

```
%patch('Vertices',verts,'Faces',faces,'FaceVertexCData',hsv(58),'FaceColor','flat');
%view(3)
%axis equal;
%axis off
%view(52.5,30)
%view(-52.5,30)
%view(127.5,30)
%view(-127.5,30)
%view(0,90)
%view(0,270)
%view(90,0)%proa
%view(0,0)%lateral, proa para direita
%view(-90,0)%popa
%view(180,0)%lateral,proa para esquerda


%           Naval Postgraduate School - CA
%   Type    : Procedure
%   Name    : Point Sur
%   Function    : Draws Point Sur Ship Model
%   Type    : Function
%   Name    : pointSur.m
%   Function    : Creates the Point Sur Ship model
%   Date    20/may/2000
%   Version : 1.0
%   Author  : Jorge Amaral Alves, LCDR (Brazilian Navy)

%==============================
% define vertices and surfaces
%==============================

function [verts,faces]=pointsur();


verts=[93,   0, 0   %0
        81,  0, -15 %1
        39, 16, -15 %2
        -43,    16, -15 %3
        -43,-16, -15    %4
        39, -16, -15    %5
        51, -16,    0   %6
        47, -16,  -5    %7
       37, -10,  -5  %8
        37,-16,  12  %9
       17, -16,  12 %10
       15, -16,  -5 %11
       -43,-16,  -8 %12
       -43, 16,  -8 %13
       15,  16,  -5 %14
       17,  16,  12   %15
       37,  16,  12 %16
       37,  10,  -5 %17
       47,  16,  -5   %18
       51,  16, 0   %19
       21,   3,  12 %20
       17,   3,  12 %21
       17,  -3,  12 %22
       21,  -3,  12 %23
       21,   3,  19     %24
       17,   3,  19     %25
       17,  -3,  19 %26
       21,  -3,  19 %27
       6,   16,  -8 %28
       -6,  16,  -8 %29
       -6,   8,  -8 %30
       6,    8,  -8 %31
       6,   16,   6 %32
       -6,  16, 6   %33
       -6,   8, 6   %34
       6,    8, 6   %35
       -16, 16,  -8 %36
       -16, 16,  -5 %37
       -16,-16,  -5 %38
       -16,-16,  -8 %39
       38, -16, 8   %40
       38,  16, 8   %41
       31,  16, 8   %42
       31,  16,  12 %43
       31, -16,  12 %44
       31, -16, 8   %45
       17, -16,  -8 %46
       17,  16,  -8 %47
       41, -16, 4   %48
       41,  16, 4   %49
       33,  16, 4   %50
       33, -16, 4   %51
       31, -16,  -5 %52
       31,  16,  -5 %53
       31,  16, 4   %54
       27,  16, 4   %55
       25,  16,  -5 %56
       29,  16,  -5 %57
       25,  16, 4   %58
       21,  16, 4   %59
       19,  16,  -5 %60
       23,  16,  -5   %61
       31, -16, 4   %62
       27, -16, 4   %63
      25, -16,  -5  %64
       29, -16,  -5 %65
```

```
      25, -16, 4   %66
      21, -16, 4   %67
      19, -16,  -5 %68
      23, -16,  -5 %69
     37,   10, 4   %70
      37, -10, 4   %71
      17, -10, 4   %72
      17, -10,  -5 %73
      17,  10,  -5 %74
      17,  10, 4]; %75

   %number =       32

   plane0  =   [0,1,2,18,19,19,19,19,19,19,19,19,19,19,19,19];
   plane1  =   [2,3,13,47,14,18,18,18,18,18,18,18,18,18,18,18];
   plane2  =   [4,5,7,11,46,12,12,12,12,12,12,12,12,12,12,12];
   plane3  =   [0,6,7,5,1,1,1,1,1,1,1,1,1,1,1,1];
   late0   =   [47,14,37,36,36,36,36,36,36,36,36,36,36,36,36,36];
   late1   =   [8,71,72,73,73,73,73,73,73,73,73,73,73,73,73,73];
   late2   =   [17,70,75,74,74,74,74,74,74,74,74,74,74,74,74,74];
   late3   =   [52,51,48,40,45,44,10,11,68,67,66,69,64,63,62,65];
   late4   =   [53,50,49,41,42,43,15,14,60,59,58,61,56,55,54,53];
   bottom  =   [1,2,3,4,5,5,5,5,5,5,5,5,5,5,5,5];
   glass0  =   [48,40,41,49,49,49,49,49,49,49,49,49,49,49,49,49];
   glass1  =   [45,44,43,42,42,42,42,42,42,42,42,42,42,42,42,42];
   glass2  =   [8,17,70,71,71,71,71,71,71,71,71,71,71,71,71,71];
   radar0  =   [20,24,27,23,23,23,23,23,23,23,23,23,23,23,23,23];
   radar1  =   [20,21,25,24,24,24,24,24,24,24,24,24,24,24,24,24];
   radar2  =   [22,23,27,26,26,26,26,26,26,26,26,26,26,26,26,26];
   stack0  =   [28,32,35,31,31,31,31,31,31,31,31,31,31,31,31,31];
   stack1  =   [32,33,29,28,28,28,28,28,28,28,28,28,28,28,28,28];
   stack2  =   [30,31,35,34,34,34,34,34,34,34,34,34,34,34,34,34];
   stack3  =   [30,34,33,29,29,29,29,29,29,29,29,29,29,29,29,29];
   deck0   =   [0,19,6,6,6,6,6,6,6,6,6,6,6,6,6,6];
   deck1   =   [6,7,18,19,19,19,19,19,19,19,19,19,19,19,19,19];
   deck2   =   [7,8,17,18,18,18,18,18,18,18,18,18,18,18,18,18];
   deck3   =   [15,16,9,10,22,23,20,21,21,21,21,21,21,21,21,21];
   deck4   =   [24,25,26,27,27,27,27,27,27,27,27,27,27,27,27,27];
   deck5   =   [14,15,21,25,26,22,10,11,11,11,11,11,11,11,11,11];
   deck6 = [11,38,37,29,30,31,28,14,14,14,14,14,14,14,14,14];
   deck7   =   [36,37,38,39,39,39,39,39,39,39,39,39,39,39,39,39];
   deck8   =   [36,13,12,39,39,39,39,39,39,39,39,39,39,39,39,39];
   deck9   =   [3,4,12,13,13,13,13,13,13,13,13,13,13,13,13,13];
   deck10  =   [32,33,34,35,35,35,35,35,35,35,35,35,35,35,35,35];
   deck11  =   [40,45,42,41,41,41,41,41,41,41,41,41,41,41,41,41];

  %faces

  faces=[  plane0
       plane1
       plane2
       plane3
       late0
       late1
       late2
       late3
       late4
       bottom
       glass0
     glass1
     glass2
     radar0
     radar1
     radar2
     stack0
     stack1
     stack2
       stack3
      deck0
      deck1
      deck2
      deck3
      deck4
      deck5
      deck6
      deck7
      deck8
      deck9
     deck10
     deck11];

faces=faces+1;
```

A.2.4 Rotina de geração das vistas sintéticas

```
function gera_vistas(classe, azimute, elevacao, distancia, dir)

% Gera vistas para cada um tipo de navio especificado.  A  quantidade de vistas
% geradas depende dos comprimentos dos arrays azimute, elevacao e distancia.
%
% classe:     'aircarrier' 'frigate' 'destroyer' 'pointsur' 'merchant'.
% azimute:    array que contem os valores de azimute (graus).
% elevacao:   array que contem os valores de elevacao (graus).
% distancia:  array que contem os valores de distancia (metros) entre  o  navio
```

```
%             (CameraTarget) e a camera.
% dir:        diretorio de saida.
%
% Apos  geradas, as  vistas  sao  gravadas  em  arquivos '.jpg' e,  os dados, em
% arquivos  '.txt'  os  quais  serao  empregados na extracao das caracteristicas
% necessarias a construcao das curvas principais.
%
% Os arquivos que contem as informacoes de  patch  devem estar contidos no mesmo
% diretorio  desta  funcao.  As  vistas  e  os dados serao gravados no diretorio
% especificado na string 'dir'.

% ================================================

% Conjuntos de treino:
% classe:    A  cada  vez que o programa roda emprega-se uma dentre as seguintes
%            strings:'aircarrier','frigate','destroyer','pointsur'ou 'merchant'.
% azimute:   azimute1  = -90:5:-50; azimute2 = -45:15:45; azimute3 = 50:5:85;
%            azimute = [azimute1 azimute2 azimute3];
% elevacao:  elevacao  = [0 7 15 22 30 45];
% distancia: distancia = 2500;
% dir:       dir = ['D:\Helena\COPPE\Mestrado\Resultados\DadosTreino\' classe '\'];

% ================================================

% Conjuntos de teste:
% classe:    A  cada  vez que o programa roda emprega-se uma dentre as seguintes
%            strings:'aircarrier','frigate','destroyer','pointsur'ou 'merchant'.
% azimute:   azimute  = -90:1:89;
% elevacao:  elevacao  = 0:1:45;
% distancia: distancia = 2500;
% dir:       dir = ['D:\Helena\COPPE\Mestrado\Resultados\DadosTeste\' classe '\'];

% ================================================

classe    = char( 'aircarrier' , 'destroyer' , 'frigate', 'merchant' , 'pointsur' );
azimute   = -90:1:89;
elevacao  = 0:1:45;
distancia = 2500;

% ================================================

for c = 1:size(classe,1)

    clf;
    pause(1);

    dir = ['D:\Helena\COPPE\Mestrado\Resultados\DadosTeste\' ...
           deblank(classe(c,:)) '\'];
    eval([ '[verts,faces]=' deblank(classe(c,:)) ';' ]);
    patch('Vertices',verts,'Faces',faces);

    camproj( 'perspective' );      % camproj {orthographic} | perspective
    ctarg = get(gca, 'CameraTarget');
    set(gca, 'CameraViewAngle', 5);

    axis equal; axis off; axis vis3d;

    arrayPontoEsq = [];

    for i = 1:length(distancia)
        for j = 1:length(azimute)
            for k=1:length(elevacao)

                [x, y, z] = calcula_coordenadas(distancia(i),azimute(j), ...
                                                elevacao(k),ctarg);
                set(gca, 'CameraPosition', [x y z]);

                [X,map]=capture;
                X=X-65;     % X so possui valores 0 (fundo) e 1 (vista)

                pause(1);

                tag = ['_d' num2str(distancia(i)) '_a' num2str(azimute(j)) ...
                       '_e' num2str(elevacao(k))];
                %filename = [dir  deblank(classe(c,:))  tag '.jpg'];
                %imwrite(X, filename, 'jpg');

                filename = [dir  deblank(classe(c,:)) tag ];
                save(filename, 'X');

            end;
        end;
    end;
end;


function [x, y, z] = calcula_coordenadas(d, az, el, ctarg)

az = deg2rad(az);  el = deg2rad(el);

x =  d * cos(el) * sin(az);
y = -d * cos(el) * cos(az);
z =  d * sin(el);

x = x - ctarg(1);
y = y - ctarg(2);
z = z - ctarg(3);
```

A.2.5 Rotina de mapeamento do tipo I

```
% ==========================================================
% Rotina processa_ConjTreinoSint_T1.m
% ==========================================================

classe1 = 'aircarrier'; classe2 = 'destroyer';
classe3 = 'frigate'; classe4 = 'merchant'; classe5 = 'pointsur';
n_classes = 5;

dir = ['D:\Helena\COPPE\Mestrado\Resultados\DadosTreino\' classe '\'];

matriz_cotas = [];

distancia = 2500;
azimute1  = -90:5:-50; azimute2 = -45:15:45; azimute3 = 50:5:85;
azimute = [azimute1 azimute2 azimute3];
elevacao  = [0 7 15 22 30 45];

c=1;
for h = 1:n_classes
  eval(['classe = classe' num2str(h) ';']);
  dir = ['D:\Helena\COPPE\Mestrado\Resultados\DadosTeste\' classe '\'];
  for i = 1:length(distancia)
    for j = 1:length(azimute)
        for k=1:length(elevacao)
            tag = ['_d' num2str(distancia(i)) '_a' num2str(azimute(j)) ...
                   '_e' num2str(elevacao(k)) 'F'];
            filename = [dir  classe  tag '.mat'];
            load(filename);

            cota = obtem_cotasDadosSint(X);
            for w=1:size(cota,2)
                matriz_cotas(c,w)=cota(w);
            end

            disp(['cota # ' num2str(c)]);
            c=c+1;
        end
    end
  end
  tag = 'cotasTreino';
  filename = [dir  classe tag '.txt'];
  save(filename, 'matriz_cotas', '-ASCII');
end


% ==========================================================
% Rotina processa_ConjTesteSint_T1.m
% Extrai cotas do tipo 1 de cada uma das vistas dos conjuntos de teste.
% ==========================================================

classe1 = 'aircarrier'; classe2 = 'destroyer';
classe3 = 'frigate'; classe4 = 'merchant'; classe5 = 'pointsur';
n_classes = 5;

dir = ['D:\Helena\COPPE\Mestrado\Resultados\DadosTeste\' classe '\'];

matriz_cotas = [];

distancia = 2500;
azimute   = -90:1:89;
elevacao  = 0:1:45;

msg = [];

for h = 1:n_classes
  eval(['classe = classe' num2str(h) ';']);
  dir = ['D:\Helena\COPPE\Mestrado\Resultados\DadosTeste\' classe '\'];
  c=1;
  for i = 1:length(distancia)
    for j = 1:length(azimute)
        for k = 1:length(elevacao)

            tag = ['_d' num2str(distancia(i)) '_a' num2str(azimute(j)) '_e' ...
                   num2str(elevacao(k)) ];
            filename = [dir  classe  tag '.mat'];
            load(filename);

            [cota, menor_cota, maior_cota] = obtem_cotasDadosSint_T1(X);
            for w=1:size(cota,2)
                matriz_cotas(c,w)=cota(w);
            end

            disp(['cota # ' num2str(c)]);
            c=c+1;
        end
    end
  end
  tag = 'cotasTeste';
  dir = 'D:\helena\COPPE\Mestrado\Curvas Principais\dados\cotaTipo1\';
  filename = [dir  classe tag '.txt'];
  save(filename, 'matriz_cotas', '-ASCII');
end


function [array_cotas, menor_cota, maior_cota] = obtem_cotasDadosSint_T1( X )

n_cotas   = 300;
```

```
[nlin_X , ncol_X] = size( X );
array_zeros = zeros( nlin_X , 1 );
array_todasCotas = [];

for init_col = 1:ncol_X
    if ~isequal( X(:,init_col) , array_zeros )
        break;
    end;
end;

if init_col >= ncol_X
    disp('Figura vazia.');
    return
end;

for fim_col = ncol_X:-1:init_col
    if ~isequal( X(:,fim_col) , array_zeros )
        break;
    end;
end;

for col = init_col:fim_col
    lin = 1;
    while (X(lin,col) == 0) & (lin < nlin_X)
        lin = lin + 1;
    end;
    array_todasCotas = [ array_todasCotas ; lin ];
end;

% ====================================

array_cotas(1) = array_todasCotas(1);

passo = length(array_todasCotas) / n_cotas;  % intervalo entre cotas

for indcota = 2 : n_cotas

    prox_cota = ceil( (indcota - 1) * passo );

    if prox_cota <= length(array_todasCotas)
        array_cotas(indcota) = array_todasCotas(prox_cota);
    else
        array_cotas(indcota) = array_todasCotas(length(array_cotas));
    end;

end;

% ====================================

% Normaliza cotas
menor_cota = min(array_cotas);
array_cotas = array_cotas - menor_cota;

maior_cota = max(array_cotas);
array_cotas = array_cotas / maior_cota;
```

A.2.6 Rotina de mapeamento do tipo II

```
% ============================================================
% Rotina processa_ConjTreinoSint_T2.m
% ============================================================

% Rotina de obtencao de cotas do tipo 2: cotas  horizontais que
% mapeiam o lado direito ou  esquerdo, dependendo da inclinacao
% da vista; no caso  de  dados  sinteticos, medida  apenas pela
% posicao relativa do ponto A2.

% Helena L. Fernandez - 17/07/2004.

global X

classe1 = 'aircarrier'; classe2 = 'destroyer'; classe3 = 'frigate';
classe4 = 'merchant'; classe5 = 'pointsur';
n_classes = 5;

matriz_cotas = [];

distancia = 2500;
azimute1  = -90:5:-50; azimute2 = -45:15:45; azimute3 = 50:5:85;
azimute = [azimute1 azimute2 azimute3];
elevacao  = [0 7 15 22 30 45];

for h = 1:n_classes
    eval(['classe = classe' num2str(h) ';']);
    dir = ['D:\Helena\COPPE\Mestrado\Resultados\DadosTreino\' classe '\'];
    c=1;
    for i = 1:length(distancia)
        for j = 1:length(azimute)
            for k=1:length(elevacao)
                tag = ['_d' num2str(distancia(i)) '_a' num2str(azimute(j)) ...
                       '_e' num2str(elevacao(k)) 'F'];
                filename = [dir  classe  tag '.mat'];
                load(filename);

                cota = obtem_cotasDadosSint_T2;
```

```
                for w=1:size(cota,2)
                    matriz_cotas(c,w)=cota(w);
                end;

                disp(['cota # ' num2str(c)]);
                c=c+1;
            end
        end
    end
    tag = 'cotasT2_Treino';
    dir = ['D:\Helena\COPPE\Mestrado\Curvas Principais\dados\cotaTipo2\'];
    filename = [dir  classe '_' tag '.txt'];
    save(filename, 'matriz_cotas', '-ASCII');
end


% =========================================================
% Rotina processaConjTesteSint_T2.m
% =========================================================

classe1 = 'aircarrier'; classe2 = 'destroyer';
classe3 = 'frigate'; classe4 = 'merchant'; classe5 = 'pointsur';
n_classes = 5;

global X;

matriz_cotas = [];

distancia = 2500;
azimute  = -90:1:89;
elevacao  = 0:1:45;

msg = [];

for h = 1:n_classes
    eval(['classe = classe' num2str(h) ';']);
    dir = ['D:\Helena\COPPE\Mestrado\Resultados\DadosTeste\' classe '\'];
    c=1;
    for i = 1:length(distancia)
        for j = 1:length(azimute)
            for k = 1:length(elevacao)
                tag = ['_d' num2str(distancia(i)) '_a' num2str(azimute(j)) '_e' ...
                        num2str(elevacao(k)) ];
                filename = [dir  classe  tag '.mat'];
                load(filename);

                cota = obtem_cotasDadosSint_T2;
                for w=1:size(cota,2)
                    matriz_cotas(c,w)=cota(w);
                end;

                disp(['cota # ' num2str(c)]);
                c=c+1;
            end
        end
    end
    tag = 'cotasT2_Teste';
    dir = ['D:\Helena\COPPE\Mestrado\Curvas Principais\dados\cotaTipo2\'];
    filename = [dir  classe '_' tag '.txt'];
    save(filename, 'matriz_cotas', '-ASCII');
end


function array_cotas = obtem_cotasDadosSint_T2
% ====================================================================
% Obtem  apenas  cotas  horizontais do trecho  lateral  compreendido
% entre os pontos A3 e A2 ou A1 e A2, conforme a posicao de A2. Se o
% valor de coluna do ponto A2 esta mais para a direita, varre-se  de
% A3 para A2; senao, de A1 para A2.
% ====================================================================

global X

n_cotas = 300;
[nlin_X, ncol_X] = size(X);

% ====================================================================
% P1 e o ponto mais acima, de onde se obtem a primeira cota.  P2 e o
% ponto mais abaixo, de onde se obtem a ultima cota. A variavel lado
% contem o valor 'dir' ou 'esq'.
% ====================================================================

[ P1, P2, lado ] = obtem_limites_T2;
%disp(['P1 = (' num2str(P1(1)) ',' num2str(P1(2)) ')']);
%disp(['P2 = (' num2str(P2(1)) ',' num2str(P2(2)) ')']);
%disp(['lado = ' lado]);

% ====================================================================
% Obtem as cotas horizontais.
% ====================================================================

array_cotas = [];

passo = ( P2(1) - P1(1) + 1 ) / n_cotas;

lin = P1(1);
prox_lin = P1(1);

for indcota = 1 : n_cotas
    if lado == 'esq'
```

```
        if lin <= P2(1)
            col = 1;
            while (X(lin,col) == 0) & (col < ncol_X)
                col = col + 1;
            end;
            array_cotas(indcota) = col;
        else
            array_cotas(indcota) = array_cotas(indcota - 1);
        end;
    else
        if lin <= P2(1)
            col = ncol_X; cota = 1;
            while (X(lin,col) == 0) & (col > 1)
                col = col - 1;
                cota = cota + 1;
            end;
            array_cotas(indcota) = cota;
        else
            array_cotas(indcota) = array_cotas(indcota - 1);
        end;
    end;
    prox_lin = prox_lin + passo;
    lin = floor(prox_lin);
end;

% ================================================================
% Normaliza os valores das cotas horizontais.
% ================================================================

array_cotas = array_cotas - ( min(array_cotas) );

% Ha casos onde a lateral do navio e´ plana (veja, por ex, a vista do
% destroyer  para  azimute = -90  e  elevacao = 0).  Neste  caso, os
% valores de cota sao todos iguais a 1 e, apos  o  primeiro passo da
% normalizacao, o valor maximo de cota sera igual a zero.
if max(array_cotas) > 0
    array_cotas = array_cotas / max(array_cotas);
end;
```

A.2.7 Rotina de mapeamento do tipo III

```
% ============================================================
% script processa_ConjTreinoSint_T3.m
% ============================================================

% Rotina de  obtencao de cotas do tipo 3 (cotas  verticais  que
% mapeiam o contorno superior da vista e cotas  horizontais que
% mapeiam o lado direito ou  esquerdo, dependendo da inclinacao
% da vista (no caso  de  dados  sinteticos, medida  apenas pela
% posicao relativa do ponto A2).

% Helena L. Fernandez - 17/07/2004.

global X

classe1 = 'aircarrier'; classe2 = 'destroyer'; classe3 = 'frigate';
classe4 = 'merchant'; classe5 = 'pointsur';

n_classes = 5;

matriz_cotas = [];

distancia = 2500;
azimute1  = -90:5:-50; azimute2 = -45:15:45; azimute3 = 50:5:85;
azimute = [azimute1 azimute2 azimute3];
elevacao  = [0 7 15 22 30 45];


for h = 1:n_classes
    eval(['classe = classe' num2str(h) ';']);
    dir = ['D:\Helena\COPPE\Mestrado\Resultados\DadosTreino\' classe '\'];
    c=1;
    for i = 1:length(distancia)
        for j = 1:length(azimute)
            for k=1:length(elevacao)
                tag = ['_d' num2str(distancia(i)) '_a' num2str(azimute(j)) ...
                       '_e' num2str(elevacao(k)) 'F'];
                filename = [dir  classe  tag '.mat'];
                load(filename);

                cota = obtem_cotasDadosSint_T3;
                for w=1:size(cota,2)
                    matriz_cotas(c,w)=cota(w);
                end
                disp(['cota # ' num2str(c)]);
                c=c+1;
            end
        end
    end
    tag = 'cotasTreino';
    dir = ['D:\Helena\COPPE\Mestrado\Curvas Principais\dados\cotaTipo3\'];
    filename = [dir  classe '_' tag '.txt'];
    save(filename, 'matriz_cotas', '-ASCII');
end
```

```
% ============================================================
% Rotina processa_ConjTesteSint_T3.m

% Rotina de  obtencao de cotas do tipo 3 (cotas  verticais  que
% mapeiam o contorno superior da vista e cotas  horizontais que
% mapeiam o lado direito ou  esquerdo, dependendo da inclinacao
% da vista (no caso  de  dados  sinteticos, medida  apenas pela
% posicao relativa do ponto A2).

% Helena L. Fernandez - 21/07/2004.
% ============================================================

global X

classe1 = 'aircarrier'; classe2 = 'destroyer'; classe3 = 'frigate';
classe4 = 'merchant'; classe5 = 'pointsur';
n_classes = 5;

distancia = 2500;
azimute   = -90:1:89;
elevacao  = 0:1:45;

matriz_cotas = [];

for h = 5:n_classes
    eval(['classe = classe' num2str(h) ';']);
    dir = ['D:\Helena\COPPE\Mestrado\Resultados\DadosTeste\' classe '\'];
    c=1;
    for i = 1:length(distancia)
        for j = 1:length(azimute)
            for k=1:length(elevacao)
                tag = ['_d' num2str(distancia(i)) '_a' num2str(azimute(j)) ...
                        '_e' num2str(elevacao(k))];
                filename = [dir  classe  tag '.mat'];
                load(filename);

                cota = obtem_cotasDadosSint_T3;
                for w=1:size(cota,2)
                    matriz_cotas(c,w)=cota(w);
                end
                disp(['cota # ' num2str(c)]);
                c=c+1;
            end
        end
    end
    tag = 'cotasTeste';
    dir = ['D:\Helena\COPPE\Mestrado\Curvas Principais\dados\cotaTipo3\'];
    filename = [dir  classe '_' tag '.txt'];
    save(filename, 'matriz_cotas', '-ASCII');
end


function array_cotas = obtem_cotasDadosSint_T3

% ================================================================
% Obtem  cotas  verticais  no  trecho  entre  os  pontos  A3 e  A1 e
% horizontais no trecho  lateral  compreendido entre os pontos  A3 e
% A2 ou A1 e A2, conforme a posicao de A2. Se o valor  de  coluna do
% ponto A2 esta mais para a  direita, varre-se de A3 para A2; senao,
% de A1 para A2.
% ================================================================

global X

n_cotas = 400;
[nlin_X, ncol_X] = size(X);

% ================================================================
% Extrai pontos A1, A2 e A3 (veja Jain, p.393).
% ================================================================

[ A1, A2, A3, lado ] = obtem_limites_T3;

% ================================================================
% Calcula o numero de cotas em cada direcao.
% ================================================================

a = A1(2) - A3(2) + 1;
if lado == 'esq'
    b = A2(1) - A3(1) + 1;
else
    b = A2(1) - A1(1) + 1;
end;

n_cotas_vert = round( n_cotas * ( a / (a + b) ) );
if rem(n_cotas_vert,2) == 1
    n_cotas_vert = n_cotas_vert + 1;
end;
n_cotas_horiz = n_cotas - n_cotas_vert;

% ================================================================
% Obtem as cotas verticais.
% ================================================================

array_cotas_vert = [];
passo = a / n_cotas_vert;
col = A3(2);
prox_col = col;

for indcota = 1 : n_cotas_vert
```

```
    if  col <= A1(2)
        lin = 1;
        while (X(lin,col) == 0) & (lin < nlin_X)
            lin = lin + 1;
        end;
        array_cotas_vert(indcota) = lin;
    else
        array_cotas_vert(indcota) = array_cotas_vert(indcota -1);
    end;
    prox_col = prox_col + passo;
    col = floor(prox_col);
end;


% ================================================================
% Obtem as cotas horizontais.
% ================================================================

array_cotas_horiz = [];
passo = b / n_cotas_horiz;
if lado == 'esq'
    lin = A3(1);
else
    lin = A1(1);
end;
prox_lin = lin;

for indcota = 1 : n_cotas_horiz
    if lado == 'esq'
        if lin <= A2(1)
            col = 1;
            while (X(lin,col) == 0) & (col < ncol_X)
                col = col + 1;
            end;
            array_cotas_horiz(indcota) = col;
        else
            array_cotas_horiz(indcota) = array_cotas_horiz(indcota - 1);
        end;
    else
        if lin <= A2(1)
            col = ncol_X;
            cota = 1;
            while (X(lin,col) == 0) & (col > 1)
                col = col - 1;
                cota = cota + 1;
            end;
            array_cotas_horiz(indcota) = cota;
        else
            array_cotas_horiz(indcota) = array_cotas_horiz(indcota - 1);
        end;
    end;
    prox_lin = prox_lin + passo;
    lin = floor(prox_lin);
end;

% ================================================================
% Normaliza as cotas verticais.
% ================================================================

array_cotas_vert  = array_cotas_vert - ( min(array_cotas_vert) );
array_cotas_vert  = array_cotas_vert / ( max(array_cotas_vert) );

% ================================================================
% Normaliza as cotas horizontais.
% ================================================================

array_cotas_horiz = array_cotas_horiz - ( min(array_cotas_horiz) );

% Ha casos onde a lateral do navio e plana (veja, por ex, a vista do
% destroyer  para  azimute = -90  e  elevacao = 0).  Neste  caso, os
% valores de cota sao todos iguais a 1 e, apos  o  primeiro passo da
% normalizacao, o valor maximo de cota sera igual a zero.

if max(array_cotas_horiz) > 0
    array_cotas_horiz = array_cotas_horiz / max(array_cotas_horiz);
end;

% ================================================================
% Concatena os arrays de cotas.
% ================================================================

array_cotas = [array_cotas_vert  array_cotas_horiz];
```

A.2.8 Rotina de mapeamento do tipo IV

```
% ========================================================
% script processa_ConjTreinoSint_T4.m
% ========================================================

% Rotina de obtencao de cotas do tipo 4.
% Helena L. Fernandez - 22/07/2004.

global X

classe1 = 'aircarrier'; classe2 = 'destroyer'; classe3 = 'frigate'; classe4 = 'merchant'; classe5 = 'pointsur';
```

```
n_classes = 5;

matriz_cotas = [];

distancia = 2500;
azimute1  = -90:5:-50; azimute2 = -45:15:45; azimute3 = 50:5:85;
azimute = [azimute1 azimute2 azimute3];
elevacao  = [0 7 15 22 30 45];

for h = 1:n_classes
    eval(['classe = classe' num2str(h) ';']);
    dir = ['D:\Helena\COPPE\Mestrado\Resultados\DadosTreino\' classe '\'];
    c=1;
    for i = 1:length(distancia)
        for j = 1:length(azimute)
            for k=1:length(elevacao)
                tag = ['_d' num2str(distancia(i)) '_a' num2str(azimute(j)) ...
                        '_e' num2str(elevacao(k)) 'F'];
                filename = [dir  classe  tag '.mat'];
                load(filename);

                cota = obtem_cotasDadosSint_T4;
                for w=1:size(cota,2)
                    matriz_cotas(c,w)=cota(w);
                end

                disp(['cota # ' num2str(c)]);
                c=c+1;
            end
        end
    end
    tag = 'cotasT4_Treino';
    dir = ['D:\Helena\COPPE\Mestrado\Curvas Principais\dados\cotaTipo4\'];
    filename = [dir  classe '_' tag '.txt'];
    save(filename, 'matriz_cotas', '-ASCII');
end


% ============================================================
% Rotina processa_ConjTesteSint_T4.m
% ============================================================

classe1 = 'aircarrier'; classe2 = 'destroyer'; classe3 = 'frigate';
classe4 = 'merchant'; classe5 = 'pointsur';
n_classes = 5;

global X;

matriz_cotas = [];

distancia = 2500;
azimute  = -90:1:89;
elevacao  = 0:1:45;

for h = 1:n_classes
    eval(['classe = classe' num2str(h) ';']);
    dir = ['D:\Helena\COPPE\Mestrado\Resultados\DadosTeste\' classe '\'];
    c=1;
    for i = 1:length(distancia)
        for j = 1:length(azimute)
            for k = 1:length(elevacao)
                tag = ['_d' num2str(distancia(i)) '_a' num2str(azimute(j)) ...
                        '_e' num2str(elevacao(k)) ];
                filename = [dir  classe  tag '.mat'];
                load(filename);

                cota = obtem_cotasDadosSint_T4;
                for w=1:size(cota,2)
                    matriz_cotas(c,w)=cota(w);
                end

                disp(['cota # ' num2str(c)]);
                c=c+1;
            end
        end
    end
    tag = 'cotasT4_Teste';
    dir = ['D:\Helena\COPPE\Mestrado\Curvas Principais\dados\cotaTipo4\'];
    filename = [dir  classe '_' tag '.txt'];
    save(filename, 'matriz_cotas', '-ASCII');
end


%function array_cotas = obtem_cotasDadosSint_T4
% ================================================================
% Obtem cotas do tipo4. Essas cotas sao os valores (normalizados) de
% linha dos pontos de contorno superior obtidos entre os pontos A3 e
% A1.  O passo e obtido  atraves  do numero  de  pontos  de contorno
% extraidos. Assim, o mapeamento  nao  tem  relacao  com a  dimensao
% horizontal como na cota do tipo1.
% ================================================================

global X  C

n_cotas = 300;

[nlin_X, ncol_X] = size(X);

% ================================================================
% Extrai pontos A1 e A3 (veja Jain, p.393).
```

```
% ================================================================

[ A1, A3 ] = obtem_limites_T4;

% ================================================================
% Extrai os pontos de contorno  de interesse  (pontos de contorno da
% parte superior da  vista, contidos  entre os  pontos A3 e A1) e os
% armazena na variavel global C.
% ================================================================

n_pontos = extrai_contorno_T4(A3, A1);

% ================================================================
% Obtem cotas.
% ================================================================

array_cotas = [];
passo = n_pontos / n_cotas;
indCota = 1;
pont = 1;

for indCota = 1 : n_cotas
    array_cotas(indCota) = C(pont,1);
    pont = ceil( indCota * passo );
    if pont > n_pontos, pont = n_pontos; end
end

% ================================================================
% Normaliza cotas.
% ================================================================

array_cotas = array_cotas - ( min(array_cotas) );
array_cotas = array_cotas / ( max(array_cotas) );


function [A1, A3] = obtem_limites_T4;

global X

[nlin_X , ncol_X] = size( X );

% ====================================

array_zeros = zeros(nlin_X,1);
for col_A3 = 1:ncol_X
    if ~isequal( X(:,col_A3) , array_zeros )
        break;
    end;
end;
if col_A3 >= ncol_X
    disp('Figura vazia.');
    return
end;
for lin_A3 = 1:nlin_X
    if X(lin_A3,col_A3) == 1
        break;
    end;
end;
A3 = [lin_A3 col_A3];

% ====================================

for col_A1 = ncol_X:-1:col_A3
    if ~isequal( X(:,col_A1) , array_zeros )
        break;
    end;
end;
for lin_A1 = nlin_X:-1:1
    if X(lin_A1,col_A1) == 1
        break;
    end;
end;
A1 = [lin_A1 col_A1];


function nPoints = extrai_contorno_T4(A3, A1);

 % ============================================================
 % Extrai os pontos de contorno contidos entre A3 e A1 e conta o
 % numero de pontos  obtidos. Como  os  pontos do contorno podem
 % ser obtidos  mais de uma vez, antes de  incluir  um  ponto na
 % tabela  C, verifica-se  se  o  ponto ja existe. Os pontos sao
 % extraidos no sentido horario.
 % O algoritmo nao trata o caso da vista tocar qq borda.
 % ============================================================

global X  C

C = [];
C(1,:) = A3;
nPoints = 1;
CurrPos = A3;
OldPos(1)=A3(1);
OldPos(2)=A3(2)-1;

while 1

    if X(CurrPos(1),CurrPos(2))==1

        NewPos=turnLeft(CurrPos,OldPos);
```

```
        if X(NewPos(1),NewPos(2))==0
            if isempty( find( C(:,1)==CurrPos(1) & C(:,2)==CurrPos(2) ) )
                nPoints = nPoints + 1;
                C(nPoints,:) = CurrPos;
            end
        end

    else

        NewPos=turnRight(CurrPos,OldPos);
        if X(NewPos(1),NewPos(2))==1
           if isempty( find( C(:,1)==NewPos(1) & C(:,2)==NewPos(2) ) )
               nPoints = nPoints + 1;
               C(nPoints,:) = NewPos;
           end
        end
    end

    OldPos=CurrPos;
    CurrPos=NewPos;

    if isequal(CurrPos,A1)  % se A1 -> A3, obtem todos os pontos do contorno
        break;
    end
end


function [NewPos]= turnLeft(CurrPos, OldPos)

if(CurrPos(2) == (OldPos(2)+1))     %situacao A
    NewPos(1)=CurrPos(1)-1;
    NewPos(2)=CurrPos(2);
elseif(CurrPos(1) == (OldPos(1)+1)) %situacao B
    NewPos(1)=CurrPos(1);
    NewPos(2)=CurrPos(2)+1;
elseif(CurrPos(2) == (OldPos(2)-1)) %situacao C
    NewPos(1)=CurrPos(1)+1;
    NewPos(2)=CurrPos(2);
elseif(CurrPos(1) == (OldPos(1)-1)) %situacao D
    NewPos(1)=CurrPos(1);
    NewPos(2)=CurrPos(2)-1;
else
    error('Erro em turnLeft');
end


function [NewPos]= turnRight(CurrPos, OldPos)

if(CurrPos(2) == (OldPos(2)+1))     %situacao A
    NewPos(1)=CurrPos(1)+1;
    NewPos(2)=CurrPos(2);
elseif(CurrPos(1) == (OldPos(1)+1)) %situacao B
    NewPos(1)=CurrPos(1);
    NewPos(2)=CurrPos(2)-1;
elseif(CurrPos(2) == (OldPos(2)-1)) %situacao C
    NewPos(1)=CurrPos(1)-1;
    NewPos(2)=CurrPos(2);
elseif(CurrPos(1) == (OldPos(1)-1)) %situacao D
    NewPos(1)=CurrPos(1);
    NewPos(2)=CurrPos(2)+1;
else
    error('Erro em turnRight');
end
```

A.2.9 Rotina de mapeamento do tipo V

```
% script processa_ConjTreinoSint_T5.m

% Rotina de obtencao de cotas do tipo 5.
% Helena L. Fernandez - 22/07/2004.

global X

classe1 = 'aircarrier'; classe2 = 'destroyer'; classe3 = 'frigate';
classe4 = 'merchant'; classe5 = 'pointsur';
n_classes = 5;

matriz_cotas = [];

distancia = 2500;
azimute1  = -90:5:-50; azimute2 = -45:15:45; azimute3 = 50:5:85;
azimute = [azimute1 azimute2 azimute3];
elevacao  = [0 7 15 22 30 45];

for h = 1:n_classes
    eval(['classe = classe' num2str(h) ';']);
    dir = ['D:\Helena\COPPE\Mestrado\Resultados\DadosTreino\' classe '\'];
    c=1;
    for i = 1:length(distancia)
        for j = 1:length(azimute)
            for k=1:length(elevacao)
                tag = ['_d' num2str(distancia(i)) '_a' num2str(azimute(j)) ...
                       '_e' num2str(elevacao(k)) 'F'];
                filename = [dir  classe  tag '.mat'];
                load(filename);
```

```
                cota = obtem_cotasDadosSint_T5;
                for w=1:size(cota,2)
                    matriz_cotas(c,w)=cota(w);
                end

                disp(['cota # ' num2str(c)]);
                c=c+1;
            end
        end
    end
    tag = 'cotasT4_Treino';
    dir = ['D:\Helena\COPPE\Mestrado\Curvas Principais\dados\cotaTipo5\'];
    filename = [dir  classe '_' tag '.txt'];
    save(filename, 'matriz_cotas', '-ASCII');
end


% ============================================================
% Rotina processa_ConjTesteSint_T5.m
% ============================================================

classe1 = 'aircarrier'; classe2 = 'destroyer'; classe3 = 'frigate';
classe4 = 'merchant'; classe5 = 'pointsur';

n_classes = 5;

global X;

matriz_cotas = [];

distancia = 2500;
azimute  = -90:1:89;
elevacao  = 0:1:45;


for h = 1:n_classes
    eval(['classe = classe' num2str(h) ';']);
    dir = ['D:\Helena\COPPE\Mestrado\Resultados\DadosTeste\' classe '\'];
    c=1;
    for i = 1:length(distancia)
        for j = 1:length(azimute)
            for k = 1:length(elevacao)
                tag = ['_d' num2str(distancia(i)) '_a' num2str(azimute(j)) ...
                       '_e' num2str(elevacao(k)) ];
                filename = [dir  classe  tag '.mat'];
                load(filename);

                cota = obtem_cotasDadosSint_T5;
                for w=1:size(cota,2)
                    matriz_cotas(c,w)=cota(w);
                end

                disp(['cota # ' num2str(c)]);
                c=c+1;
            end
        end
    end
    tag = 'cotasT2_Teste';
    dir = ['D:\Helena\COPPE\Mestrado\Curvas Principais\dados\cotaTipo5\'];
    filename = [dir  classe '_' tag '.txt'];
    save(filename, 'matriz_cotas', '-ASCII');
end


function array_cotas = obtem_cotasDadosSint_T5
% ================================================================
% Obtem cotas do tipo4. Essas cotas sao os valores (normalizados) de
% linha dos pontos de contorno superior obtidos entre os pontos A3 e
% A1.  O passo e obtido  atraves  do numero  de  pontos  de contorno
% extraidos. Assim, o mapeamento  nao  tem  relacao  com a  dimensao
% horizontal como na cota do tipo1.
% ================================================================

global X  C

n_cotas = 300;

[nlin_X, ncol_X] = size(X);

% ================================================================
% Obtem o pontos A3 (veja Jain, p.393).
% ================================================================

array_zeros = zeros(nlin_X,1);
for col_A3 = 1:ncol_X
    if ~isequal( X(:,col_A3) , array_zeros )
        break;
    end;
end;
for lin_A3 = 1:nlin_X
    if X(lin_A3,col_A3) == 1
        break;
    end;
end;
A3 = [lin_A3 col_A3];

% ================================================================
% Extrai todos os pontos de contorno, iniciando por A3 e os armazena
% na variavel global C.
% ================================================================
```

```
n_pontos = extrai_contorno_T5(A3);
clear X;

% ================================================================
% Calcula o centro de massa.
% ================================================================

soma_lin = 0;
soma_col = 0;
for i = 1:n_pontos
    soma_lin = soma_lin + C(i,1);
    soma_col = soma_col + C(i,2);
end;
mlin = soma_lin / n_pontos;  % n da fig. 9.38(a)
mcol = soma_col / n_pontos;  % m da fig. 9.38(a)
cm = [ mlin mcol ];

% ================================================================
% Calcula distancias do centro de massa aos pontos do contorno.
% ================================================================

CM = repmat(cm, n_pontos, 1);
d = sqrt(sqdist(CM', C'));  d = d(1,:);

% ================================================================
% Obtem cotas.
% ================================================================

array_cotas = [];
passo = n_pontos / n_cotas;
indCota = 1;
pont = 1;

for indCota = 1 : n_cotas
    array_cotas(indCota) = d(pont);
    pont = ceil( indCota * passo );
    if pont > n_pontos, pont = n_pontos; end
end

% ================================================================
% Normaliza cotas.
% ================================================================

array_cotas = array_cotas - ( min(array_cotas) );
array_cotas = array_cotas / ( max(array_cotas) );


function nPoints = extrai_contorno_T4(A3);

 % ============================================================
 % Extrai todos os pontos de contorno e conta o numero de pontos
 % obtidos. Como os pontos do contorno podem ser obtidos mais de
 % uma vez, antes de incluir um ponto na tabela C, verifica se o
 % ponto ja existe. Os pontos sao extraidos no sentido horario.
 % O algoritmo nao trata o caso da vista tocar qq borda.
 % ============================================================

global X  C

C = [];
C(1,:) = A3;
nPoints = 1;
CurrPos = A3;
OldPos(1)=A3(1);
OldPos(2)=A3(2)-1;

while 1

    if X(CurrPos(1),CurrPos(2))==1

        NewPos=turnLeft(CurrPos,OldPos);
        if X(NewPos(1),NewPos(2))==0
            if isempty( find( C(:,1)==CurrPos(1) & C(:,2)==CurrPos(2) ) )
                nPoints = nPoints + 1;
                C(nPoints,:) = CurrPos;
            end
        end

    else

        NewPos=turnRight(CurrPos,OldPos);
        if X(NewPos(1),NewPos(2))==1
           if isempty( find( C(:,1)==NewPos(1) & C(:,2)==NewPos(2) ) )
               nPoints = nPoints + 1;
               C(nPoints,:) = NewPos;
           end
        end
    end

    OldPos=CurrPos;
    CurrPos=NewPos;

    if isequal(CurrPos,A3)  % condicao de parada
        break;
    end
end


function d = sqdist(a,b)
```

```
% sqdist - computes pairwise squared Euclidean distances between points

% original version by Roland Bunschoten, 1999

aa = sum(a.*a,1); bb = sum(b.*b,1); ab = a'*b;
d = abs(repmat(aa',[1 size(bb,2)]) + repmat(bb,[size(aa,2) 1]) - 2*ab);
```