Curso Bacharelado em Ciência da Computação

Universidade Estadual de Mato Grosso do Sul

CLASSIFICAÇÃO DE FOLHAS USANDO MEDIDAS INVARIANTES

Juliana Farias de Souza

Priscila Marques Kai

Professor Dr. Osvaldo Vargas Jaques (Orientador)

Curso de Ciência da Computação
Universidade Estadual de Mato Grosso do Sul

DOURADOS - MS

2014

CLASSIFICAÇÃO DE FOLHAS USANDO MEDIDAS INVARIANTES

Juliana Farias de Souza

Priscila Marques Kai

Este exemplar corresponde à redação da monografia da disciplina Projeto Final de Curso por Juliana Farias de Souza e Priscila Marques Kai, como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação.

Dourados, 30 de outubro de 2014

Prof. Dr. Osvaldo Vargas Jaques (orientador)

Dedico este trabalho aos nossos pais, irmãos que, com muito carinho e apoio, não mediram esforços para que nós chegássemos a está etapa de nossas vidas.

A todos os professores do curso de Ciência da Computação, que foram parte fundamental em nossas vidas acadêmicas.

Aos amigos e colegas pelo apoio constante nesta jornada e a todos os que estiveram próximos em todos esses anos.

“Que os vossos esforços desafiem as impossibilidades, lembrai-vos de que as grandes coisas do homem foram conquistadas do que parecia impossível.”

Charles Chaplin

AGRADECIMENTOS

Agradecemos primeiramente a Deus que permitiu que tudo isso acontecesse ao longo de nossas vidas, tanto na vida universitária, como em todos os momentos.

A Universidade Estadual de Mato Grosso do Sul, pela oportunidade de fazermos o curso.

Ao Prof. Dr. Osvaldo Vargas Jacques pela oportunidade e apoio na elaboração deste projeto.

RESUMO

O processamento de imagens digitalizadas possuem diversas etapas. Essas etapas têm por início a leitura de dados, passando pela segmentação de dados, extração de dados, convertendo assim, informações a fim de facilitar a análise para posteriormente realizar o reconhecimento de imagens, classificando-as. Esse projeto utilizará ferramentas computacionais de cálculo matemático, processamento de imagens e classificação tais como Matlab e Weka, para realizar todo o processo envolvendo operações sobre a imagem através do uso de funções e procedimentos. O processo de segmentação da imagem é feito pela técnica de binarização através do método de Otsu e o reconhecimento de contornos da imagem pelo método do Algoritmo do Ceguinho utilizando após este primeiro processamento da imagem a transformada de escala. Finalmente, os modelos de aprendizados de máquina são ferramentas importantes para a classificação, obtendo percentuais sobre acertos e erros, necessárias para a classificação.

Palavras-chave: Classificação de folhas. Segmentação. Transformada de Fourier. Análise de Componentes Principais. Aprendizado de Máquina. WEKA.

ABSTRACT

The processing of scanned images have different steps. These steps start with reading data, through by data segmentation, data extraction, converting information to facilitate the analysis and then finally to the recognition of images, classifying them. This project will use computational tools for mathematical computation, image processing and classification such as Matlab and Weka to perform the whole process involving operations over the image through the use of functions and procedures. The process of image segmentation process is made by the technique of binarization by Otsu method and the recognition of image contours by the method of Algorithm Blindie after this first image processing to transform scale method. Finally, models of machine learning are important tools for classification, percentage of getting hits and misses, necessary for classification.

Keywords: Classification of leaves. Segmentation. Fourier Transform. Principal Component Analysis. Machinelearning. Weka.

LISTA DE SIGLAS

Sigla	Significado
2D	duasdimensões
3D	trêsdimensões
GIMP	Do inglês *GNU ImageManipulationProgram*
GNU	Do inglês *GNU'snot Unix*
JPEG	do inglês *Joint Photographic Experts Group*
HD	do inglês *Hard Disk*
LMT	Do inglês *LogisticModelTrees*
PCA	do inglês *Principal ComponentAnalysis*
PDI	Processamento Digital de Imagens
RGB	modelo baseado em três cores principais. Do inglês R – red, G – green e B – blue.
TKL	Transformada Karhunen-Loève
UEMS	Universidade Estadual de Mato Grosso do Sul

LISTA DE FIGURAS

LISTA DE EQUAÇÕES

LISTA DE TABELAS

SUMÁRIO

Capítulo 1
INTRODUÇÃO

Em nosso planeta existem diversos tipos de espécies de plantas, de vários tamanhos e formas, que possuem inúmeros propósitos. Consequentemente, as folhas também apresentam variedades e dimensões diferenciadas, possuindo características particulares dependendo da espécie a que fazem parte.

Sendo a folha um importante órgão vegetal, realizando processos indispensáveis para as plantas de modo geral, estudá-las de forma a conseguir separá-las e então classificá-las é uma forma interessante de reunir informações sobre espécies de folhas conhecidas, possibilitando a catalogação das mesmas e as que possam futuramente serem descobertas.

1.1 JUSTIFICATIVA

Referente à classificação de folhas, existe atualmente variadas formas de realizá-la, baseando-se em diversos critérios como: cores, forma, nervura, habitat que residem, etc. A forma mas costumeira de classificar folhas é através do limbo, que representa a parte principal, ou seja, o formato da folha.

Figura 1.1 – Folhas de tamanhos de formas diferentes. Fonte: Wikimedia Commons.

Pelo formato de uma folha, é possível extrair demasiadas informações e reunir dados utilizando suas representações digitais. Com o auxílio de softwares de uso matemático e dando um tratamento adequado a imagem antes de realizar procedimentos de extração de dados, é possível realizar a mineração de dados e a classificação dos mesmos.

Por esse motivo, justifica-se a importância deste projeto em realizar um processo que faça classificações utilizando imagens digitais de folhas sem a necessidade de custos elevados, examinando e realizando o tratamento inicial da imagem, extraindo informações e dando continuidade através de técnicas de mineração com uso de softwares auxiliares, para então obtermos um percentual satisfatório de acertos relacionados a classificação.

1.2 OBJETIVOS DO PROJETO

A utilização de imagens digitais para análise tem sido algo corriqueiro nos dias atuais. Tendo por finalidade um método para estudo dessas imagens, o projeto foca na classificação de amostras de folhas das mais variadas espécies, realizando o cálculo dos dados com o auxílio de medidas invariantes. As imagens digitais a serem estudadas foram disponibilizadas a partir de um banco de dados de imagens digitais.

Os objetivos específicos do projeto são:

a) processamento de imagens digitais de amostras das folhas, aplicando métodos de segmentação de imagens;

b) colher dados utilizando funções de analises de componentes, extraindo informações da imagem;

c) classificar as folhas em grupos a partir dos dados obtidos.

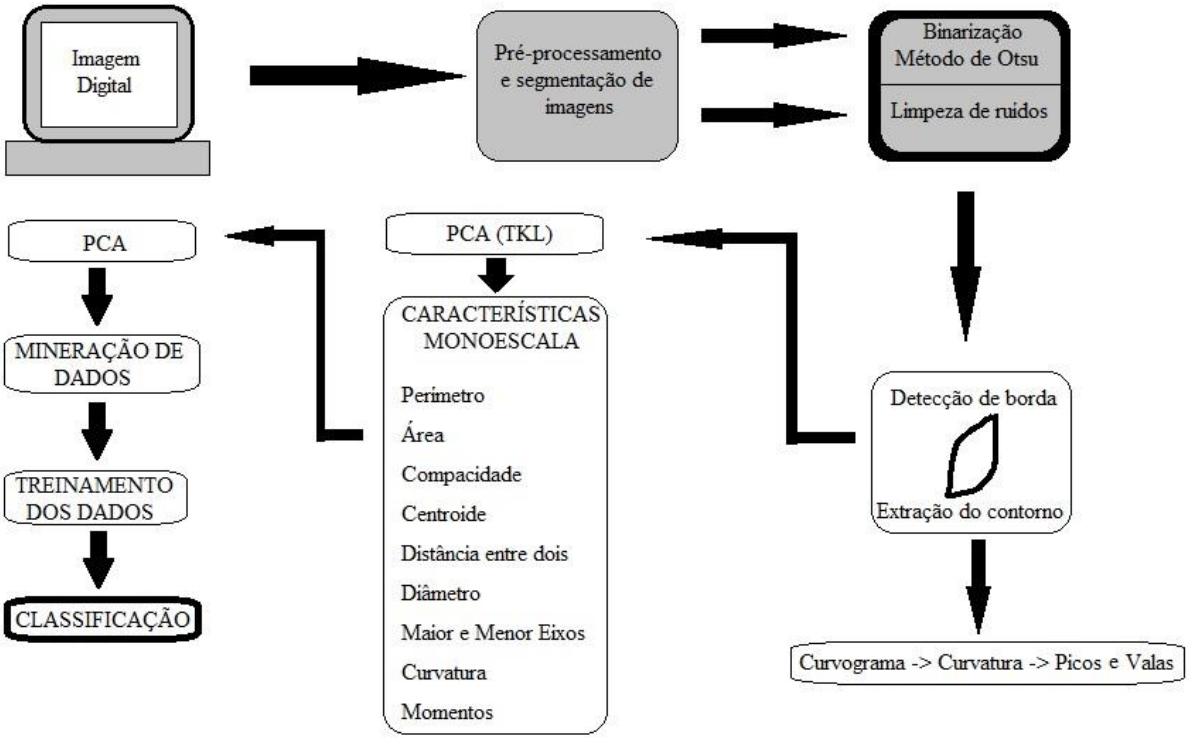


Figura 1.2. Ilustração do processo a ser desenvolvido e seu objetivo final.

1.3 ESTRUTURA DO PROJETO

O projeto em questão está dividido em quatro capítulos. Primeiramente será feita uma introdução sobre o assunto trabalhado, abordando no segundo capítulo a fundamentação teórica necessária para o entendimento do trabalho a ser realizado, introduzindo conceitos sobre formas de segmentação de imagens e as técnicas que auxiliam na extração de dados.

No terceiro capítulo será demonstrado o desenvolvimento do projeto e as ferramentas necessárias utilizadas para o trabalho, incluindo também o código desenvolvido durante todo o processo.

O quarto capítulo é composto pela apresentação de resultados obtidos do projeto.

Capítulo 2
FUNDAMENTAÇÃO TEÓRICA

Nas seções adiante, apresentamos os conceitos teóricos referentes à análise de imagens processadas digitalmente e sobre formas de obtenção de dados através de distintas funções. A seguir uma introdução sobre o processamento de imagens, na seção 2.1 e sua etapa inicial, da aquisição de imagens na seção 2.2. Na seção 2.3 temos o pré-processamento, introduzindo algumas técnicas utilizadas, como o método de binarização e a extensão da mesma por Otsu, detecção de bordas, seguindo para a etapa de extração de dados e características de imagens.

De início mostramos o processo de captura de uma imagem digital. Relataremos sobre segmentação e binarização, que transforma uma imagem RGB em escala cinza de cores, utilizando o algoritmo de Otsu, para o encontro de um limiar, para enfim fazer a transformação binária de cores de uma imagem (YANG, L. et al.2012).

Também será mostrado o processo de detecção de borda pelo método do Ceguinho, que utiliza a ideia de busca por pontos através de seus vizinhos(CESAR; COSTA, 2009). Conceitos sobre autovalores e autovetores e sobre o processo da PCA também serão explicados(VASCONCELOS, 2013).A estatística multivariada e características monoescala também serão citadas mais à frente. Assim como os modelos de aprendizagem de máquina e como funcionam.

2.1 O PROCESSAMENTO DIGITAL DE IMAGENS

As áreas de processamento de imagens e visão por computador vêm apresentando expressivo desenvolvimento nas últimas décadas. Tal crescimento pode ser detectado na área acadêmica onde o assunto é objeto de pesquisas, teses e dissertações nas mais importantes universidades brasileiras e mundiais, na esfera industrial onde a cada dia aumenta o número de empresas que produzem, comercializam e utilizam soluções de processamento eletrônico de imagens em seus processos e na vida cotidiana (MARQUES FILHO e VIEIRA NETO, 1999).

O Processamento Digital de Imagens (PDI) é o nome dado ao processo de análise que envolve a execução de tarefas, havendo a interconexão entre as mesmas, utilizando como recurso principal o computador como ferramenta de trabalho.

2.2 AQUISIÇÃO DA IMAGEM

O primeiro passo para se dar início ao processamento é a realização da captação de imagem, podendo ser feita através de diversos recursos como, por exemplo, o uso de máquinas fotográficas e scanners, que necessitarão de algum processo de digitalização posterior para então ser tratada computacionalmente e assim finalmente armazená-las em local não volátil, como cartões de memória, HDs externos, entre outros.

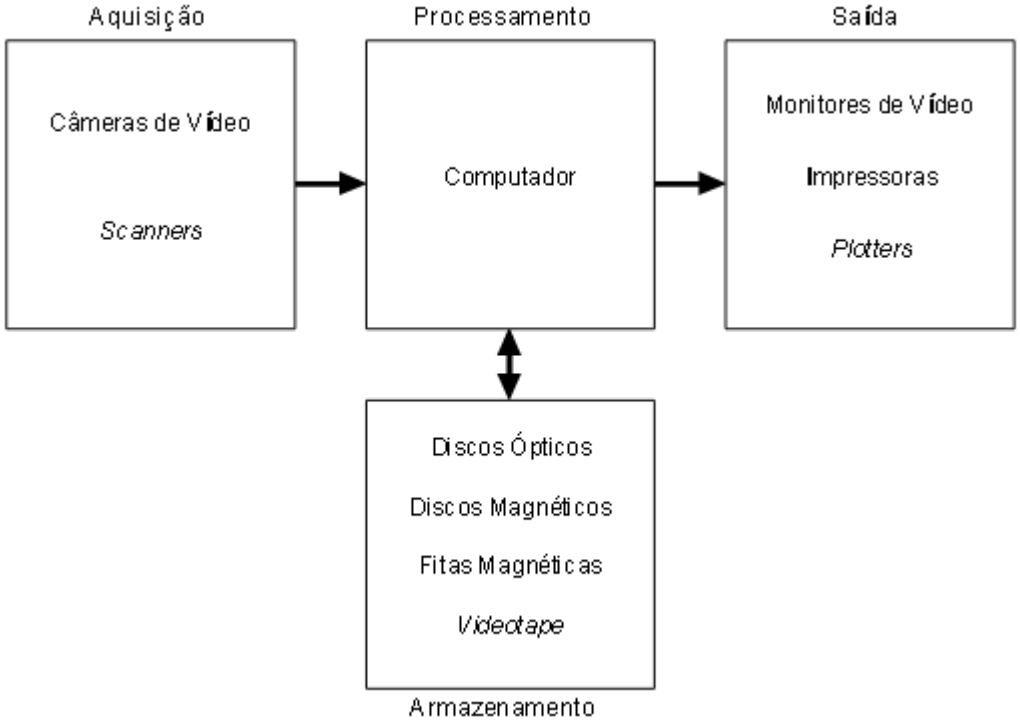


Figura 2.1 – Elementos de um sistema de processamento de imagens.
Fonte: MARQUES FILHO e VIEIRA NETO, 1999, p. 2.

A imagem obtida do dispositivo utilizado nada mais é do que o retrato de iluminação dela mesma refletida em uma superfície.

Podemos representar imagens em duas dimensões ou mais. Definindo uma imagem como uma matriz, que é composta através de uma função de duas variáveis *f(x,y)*, onde *x* e *y* correspondem a coordenadas espaciais, e *f* a intensidade da imagem em dados *x* e *y* distintos.

Dada uma matriz *f* de determinada imagem, *M* representando as linhas e *N* as colunas, cada elemento que a compõe é denominado como pixel, sendo este considerado o menor elemento de uma imagem digital.

$$f(x,y) = \begin{pmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{pmatrix}$$

Equação 2.1. Representação da imagem em uma matriz por seus pixels.

2.3 PRÉ-PROCESSAMENTO E SEGMENTAÇÃO DE IMAGENS

O processamento de imagens é envolto por muitos estágios. Em primeiro lugar realiza-se o pré-processamento, no qual é responsável pelo tratamento inicial da imagem, englobando a correção de distorções geométricas, ruídos, falhas, áreas de iluminação irregular e outros atributos que prejudicam na visualização, que consequentemente se não tratados corretamente interferem negativamente no processo de análise e extração de dados.

2.3.1 Binarização

A etapa de segmentação de imagens é um procedimento complexo, no qual problemas ocasionais de processamento de dados de uma imagem são muito comuns. Atualmente, existem diversos métodos para a análise de dados de uma imagem que utilizam dos mais variados recursos, como o uso de funções.

O ato de separar objetos é um ato rotineiro no dia a dia de profissionais e até mesmo leigos em edição de imagem. Programas para a edição de figuras, como o *Photoshop*[1], *CorelDRAW*[2], *GIMP*[3] e vários outros, oferecem recursos para a seleção de objetos, separando as partes de interesse das demais, além de outras funcionalidades.

Será abordado sobre a binarização, também denominada limiarização, que compreende no processo de converter uma dada imagem em nível de cinza para uma imagem binária, ou seja, com somente dois tons de cores: preto e branca. Este processo é classificado como um tipo específico de segmentação de imagens.

Para o processo de binarização de uma imagem são necessárias algumas etapas até de fato aplicá-la. Primeiramente é necessário identificar o objeto de interesse e separá-lo do

[1]Desenvolvido pela Adobe Systems. Versão atual: Adobe Photoshop CC (CreativeCloud, 14.0).

[2]Desenvolvidopelapor The GIMP Development Team. Licenciado sob a GNU General PublicLicense. Versão estável: GIMP 2.8.14.

[3] Desenvolvido pela Corel Corporation.Versão estável: CorelDRAW X7.

fundo da imagem, que consiste em dividir em duas classes distintas, selecionando os pixels de maior intensidade da imagem principal, apresentando o resultado em somente um bit de resolução e descartando os outros objetos.

Geralmente este processo tem como característica objetos em preto sobre um plano de fundo em branco, facilitando a identificação. O histograma é usado como base para identificar regiões com determinados picos uniformes de intensidade, sendo mais efetiva a binarização quando a imagem apresenta níveis em cinza distintos dos demais. Quanto mais diferenciáveis forem esses níveis, mais fácil é aplicado a limiarização. Entretanto, o objetivo principal deste método é a localização de um limiar considerado como ótimo, separando as duas classes principais do objeto de análise (objeto principal e o fundo).

Caso a imagem apresente pontos de luminosidade irregulares, a aplicação da binarização não terá a eficiência esperada. Neste caso, outros métodos teriam de ser utilizados em conjunto para um resultado ideal.

Para a análise de uma imagem e suas componentes, classificamos os dados como o produto de duas variáveis – iluminação e Refletância – podendo representá-las pela fórmula:

$$f(x,y) = i(x,y)R(x,y)$$

Equação 2.2. Função que calcula o produto entre a iluminação e refletância.

A função*i* representa a iluminação em coordenadas de um plano 2D, multiplicada pela refletância, correspondente a função *R*.

Podemos, matematicamente, representar a binarização como uma função que pode ser dividida em dois intervalos:

$$g(x,y) = \begin{cases} 0\ se\ f(x,y) \leq T\ como\ objeto \\ 1\ se\ f(x,y) > T\ como\ fundo \end{cases}$$

f(x,y) é o ponto em nível de cinza da imagem de entrada e variável *T* o limiar calculado. Atribui-se o valor 1 aos pixels considerados como fundo e 0 aos correspondentes à figura analisada. Se o valor da imagem de entrada for menor que o limiar, ela é classificada como fundo, caso contrário, como objeto. A função *g(x,y)* é a saída com a imagem limiarizada.

A escolha do limiar é fundamental para este processo, pois ele influencia na qualidade da limiarização aplicada. Para a seleção deste limiar são levadas em consideração algumas informações, como picos altos, simétricos ou diferenciáveis apresentados no histograma.

A binarização pode ser aplicada de duas formas: global ou localmente. No método aplicado globalmente, o limiar escolhido é um valor padrão para toda a imagem, com melhores resultados quando os níveis de cinza do objeto e o fundo são bem distintos, enquanto que na binarização local o limiar baseia de acordo com a vizinhança de pixels a partir de um ponto. Este tipo de aplicação tem melhor utilização quando os níveis de cinza do fundo e objeto não são tão diferenciáveis a ponto de poder se aplicar a binarização global. Geralmente, a escolha de um limiar global pode ser bem mais vantajosa, pois requer um processamento menor de dados.

Um limiar escolhido inadequadamente pode fazer com que todo o trabalho no processamento da imagem seja prejudicado, podendo deixar a imagem ilegível, interferindo diretamente nas próximas etapas que seriam aplicadas após a binarização. Para evitar esse problema, outros procedimentos são utilizados anteriormente a fim de realizar um pré-processamento, para daí então aplicar a binarização de forma mais eficiente.

2.3.2 Binarização pelo Método de Otsu

O Método de Otsu é uma técnica que tem por finalidade o estudo a fim de encontrar um limiar, ou seja, um valor adequado para a correta separação entre figura e fundo, considerado como ótimo de uma determinada imagem I em escala cinza de cores, convertendo os dados encontrados por meio de cores em uma nova imagem monocromática.

No método de Otsu , segundo YANG, L. et al. (2012, p.468-469), ao aplicá-lo em uma imagem com histograma diferente de unimodal, este procedimento não será realizado com mesma facilidade.

A principal função deste método é fazer com que o histograma de dada uma imagem I em escala cinza de cores, através de funções Gaussianas, seja transformado de forma a reduzir a variância entre os intervalos de classes. Cada classe é obtida pela diferença entre o limite inferior e superior de uma classe, tendo individualmente características particulares, como desvio e média.

Para a obtenção do limite inferior é necessário que a imagem analisada em questão possua um constante único em nível de cinza, enquanto que para o cálculo de limite superior são necessários dois e somente dois valores.

Uma imagem pode ser descrita como uma função intensidade em tons de cinza 2D, e contém um determinado número de pixels com níveis de cinza entre 1 a L. O número de pixels com níveis de cinza é denotado $i\ fi$ (LIAO, P.S; CHEN, T.S; CHUNG, P.C, 2001, p.

715). Supondo uma dada imagem *I*, com *M* linhas e *N* colunas, com *L* níveis de cinza, o cálculo realizado para a obtenção do histograma da imagem, é dada pela fórmula:

$$P_i = \frac{n_i}{MxN}$$

Equação 2.3. Função para cálculo do histograma.

O processo é baseado em uma análise discriminante com partições da imagem em duas classes C_0 e C_1 em nível *t* de cinza, tal que C_0 = {0, 1, 2,..., *t*} e C_1 = {*t*+1, *t*+2, *L*-1}, tal que, n_i representa o número de pixels da imagem *I* com intensidade *i* de cinza, dividida pela multiplicação do total de linhas pelas colunas, totalizando todos os pixels da imagem. São permitidas 256 intensidades de cinza, logo, i está no intervalo [1, *L*-1], *L* representando os níveis de cinza.

$$\omega(t) = \sum_{i=0}^{L-1} P_i$$

Equação 2.4. Função para calcular o limiar.

paraP_i = 1,P_i sendo ≥ 0.

C_0 e C_1 correspondem ao objeto de interesse e o plano de fundo, portanto, as definições de suas probabilidades podem ser descritas na forma:

$P_1(t)$ = probabilidade de *t* na classe C_0;

$P_2(t)$ = probabilidade de *t* na classe C_1;

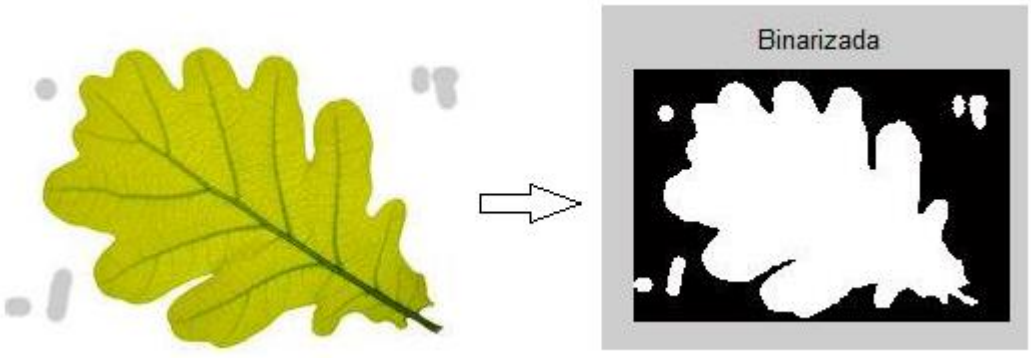


Figura 2.2. Binarização de uma folha

O método de Otsu é usado em ocasiões distintas, tais como a apresentação de histograma com grupos acumulativos de zeros, facilitando o estudo de outros aspectos da imagem e a separação entre elas, auxiliando no cálculo de níveis medianos.

2.3.3 Limpeza de ruídos

A imagem ao ser transportada de um dispositivo para o computador pode ocasionar em falhas, como a adição de ruídos, prejudicando a qualidade e visibilidade da mesma. Para a correção desses ruídos temos a técnica de limpeza de ruídos, que têm for função a filtragem e suavização de ruídos contidos em imagens, permitindo então que mais padrões sejam descobertos.

Assim como o som, a filtragem de ruídos também é um processo corriqueiro em imagens. Considerando a imagem como um sinal e analisando-a caso contenha ruído, ela implicará em frequências altas com altos valores. A solução para o tratamento da imagem, portanto, é a anulação desses picos, ou seja, utilizar um filtro que anule as regiões que possuam picos.

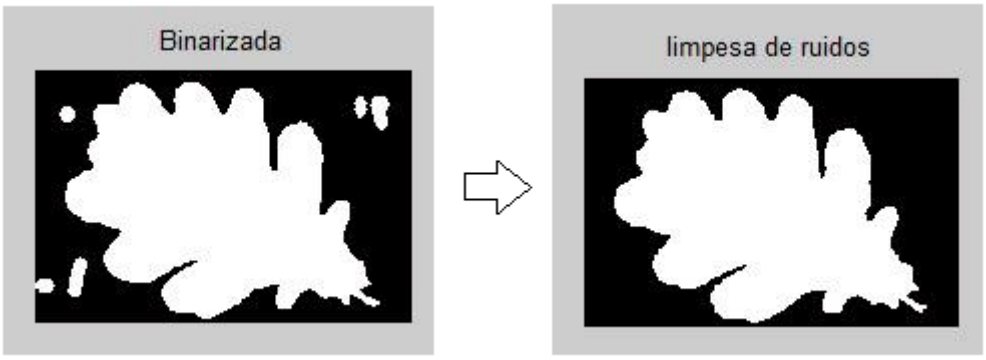


Figura 2.3. Limpeza de ruídos de uma folha

Essa etapa foi feita com a ajuda da função bwlabel do MATLAB onde ela identifica os objetos da imagem, então foi identificado o objeto maior, que no caso é a folha e eliminado os objetos restantes, resultando na imagem só a folha a ser analisada .

2.4 EXTRAÇÃO DE CARACTERÍSTICAS

Após a etapa de segmentação, o processo de extração de características nos permite a distinção dos elementos da imagem, aplicando algoritmos de reconhecimento a fim de obtermos uma saída de um conjunto de dados referentes à imagem.

As características são: Detecção de Borda utilizando o método do Ceguinho, Decomposição Espectral, Covariância, Correlação.

2.4.1 Detecção de Borda utilizando o método do Ceguinho

No processamento de imagens, a detecção de borda é utilizada para localizar pontos de uma imagem em que há discrepância no nível de intensidade da luminosidade. Mudanças

bruscas de intensidade em imagens podem revelar informações importantes, como transições entre objetos ou plano de fundo.

Para se realizar a detecção de borda de uma imagem, usa-se a representação de uma matriz 3x3.

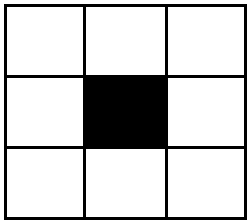


Figura 2.4 – Representação de Matriz 3x3

O ponto visualizado no centro representa o pixel de uma imagem envolto pela vizinhança de oito pontos, comparando então a diferença entre ele e seus oito pixels vizinhos.

O método do ceguinho procura pelo primeiro pixel do plano principal, assinalando-o como objeto inicial do contorno, onde após o algoritmo detectar todos os pontos pertencentes a ele, retornará um vetor contendo as coordenadas de todos os pixels do contorno, sendo este um número complexo E(i) = x +j*y.

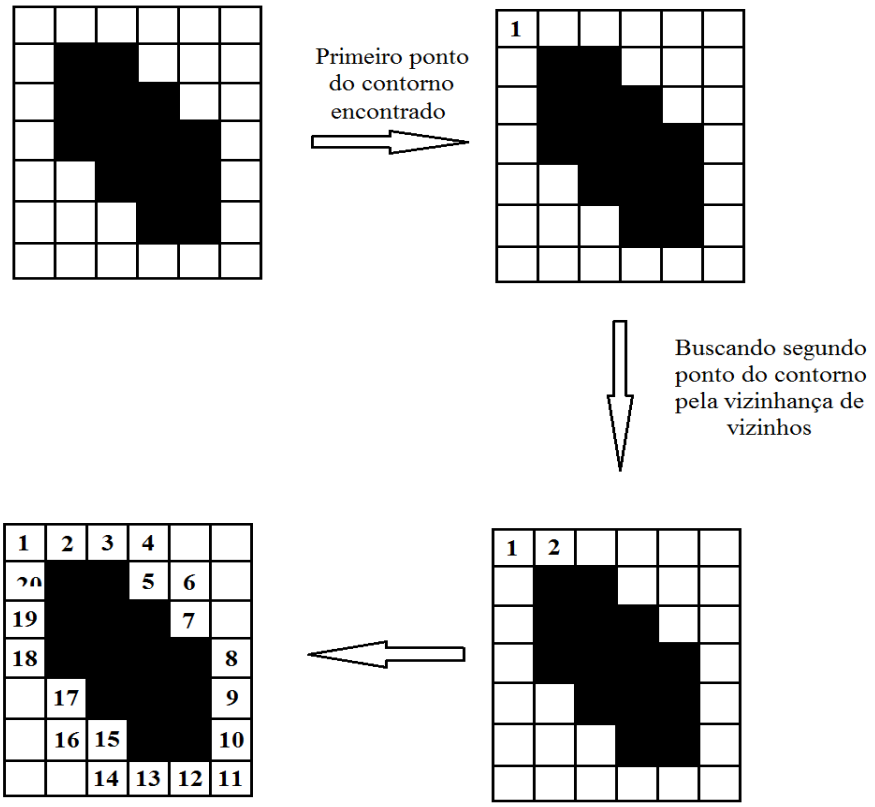


Figura 2.5. Sequência de passos do algoritmo do Ceguinho.

De modo geral, a detecção é feita através da matriz, como um distinto ponto a partir de um fundo constante, verificando a diferença entre os níveis de cinza dos pontos. Essa técnica é uma boa opção para ser utilizada para a obtenção de características de uma determinada imagem. Pois além da extração de dados, ela contribui para um menor processamento de informações, descartando as menos importantes.

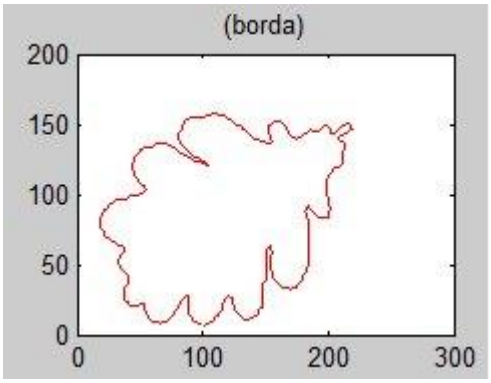


Figura 2.6. Borda de uma folha

O processo de detecção de bordas pode ser entendido como a transformação de uma imagem 2D para outra composta por curvas. Ela verifica mudanças nos tons de cinza da vizinhança de pixels.

2.4.2 Decomposição Espectral

Sendo parte fundamental da estatística multivariada, a decomposição espectral trabalha com matrizes de correlação e covariância.

É comum em metodologias e aplicações a necessidade de obter a decomposição espectral de uma matriz. Para calculá-la precisamos da obtenção de seus autovalores e autovetores, sendo estes normalizados.

O teorema de decomposição espectral diz que podemos decompor uma matriz quadrada M= $\sum_{i=1}^{n} \lambda_i \vec{e_i}\vec{e_i^t}$ sendo$\vec{e_i^t}$ o autovetor transposto de $\vec{e_i}$ que é um dos autovetores de M e λ_i o autovalores de M. Ou seja M é uma combinação linear dos autovalores e autovetores. Cada elemento desta combinação linear é uma matriz nxn, sendo nxn a dimensão de M. Basta a existência de um autovetor e autovalor para termos uma matriz nxn aplicando este teorema.

2.4.2.1 Autovalor e Autovetor

Primeiramente só e possível obter autovalores e autovetores de matrizes quadradas e o numero de autovalores e definido pela ordem da matriz.

Os autovalores de uma matriz também são chamados de valor próprio ou valor característico.

Para entendermos sua definição,segundo (FERREIRA,2012) consideremos uma matriz *A* quadrada. Ao multiplicarmos essa matriz *A* por um vetor coluna *v* não nulo, obtemos outro vetor também de dimensão *n*×1. Por outro lado, se multiplicarmos o mesmo vetor *v* por uma constante λ, também obteremos como resultado um vetor de dimensão *n*×1:

$A \cdot v$ = vetor de dimensão *n*×1

$\lambda \cdot v$ = vetor de dimensão *n*×1

$A.v = \lambda.v$onde essa constante λ é chamada de autovalores.

Portanto, autovalor é um número, real ou complexo, que de certa forma pode substituir uma matriz quadrada, ou seja, ou autovalores podem representar essa matriz.

Para a analise dos auvovalores e autovetores temos a equação:

$$Av = \lambda v$$

Essa equação ocorre em muitas aplicações de álgebra, uma das principais aplicações de autovalor é a solução de sistemas de equação diferenciais lineares.

Para achar um escalar λ usa-se o *p(λ)= determinante (A- λI)* as raízes desse polinômio característico são os autovalores de A.

Exemplo: encontre os autovetores e os autovalores associados da matriz

$$A = \begin{pmatrix} -3 & 4 \\ -1 & 2 \end{pmatrix}$$

O cálculo dos autovalores e autovetores será feito através do polinômio característico:

$$\det(A - \lambda I) = det\begin{bmatrix} -3-\lambda & 4 \\ -1 & 2-\lambda \end{bmatrix}$$

$$= (-3-\lambda)(2-\lambda) + 4$$

$$= \lambda^2 + \lambda - 2$$

$$= p(\lambda)$$

Para $p(\lambda) = 0$, será igual a:

$$(\lambda - 1)(\lambda + 2) = 0 \; ou \; \lambda = 1 \; ou \; \lambda = -2$$

Então se conclui que os autovetores de A são 1 e -2. A seguir os autovetores associados.

$\lambda = 1$ temos:

$$\begin{bmatrix} -3 & 4 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 1 \begin{bmatrix} x \\ y \end{bmatrix}$$

Logo

$$\begin{bmatrix} -3 & 4 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow \begin{cases} -4x + 4y = 0 \\ -x + y = 0 \end{cases}$$

Então podemos dizer que *x* equivale a *y*, portanto os autovetores associados a $\lambda = 1$ são os vetores $v = (x,x)$, para $x \neq 0$.

Para $\lambda = -2$ obtemos:

$$\begin{bmatrix} -3 & 4 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 2 \begin{bmatrix} x \\ y \end{bmatrix}$$

Então

$$\begin{cases} -x + 4y = 0 \; ou \; x = 4y \\ -x + 4y = 0 \end{cases}$$

Os autovetores correspondentes ao autovetor$\lambda = -2$ são da forma $v = (4y,y)$, $y \neq 0$, ou $v = (x,(1/4)x)$.

2.4.3 Covariância

A covariância entre duas variáveis pode ser descrita como amedida de duas variáveis variam conjuntamente. A covariância também é chamada muitas vezes de medida de dependência linear entre as duas variáveis aleatórias. Mede o grau de dependência linear entre duas variáveis(BOLDRINI;COSTA;FIGUEREDO;WETZLER,1980).

A covariância é calculada por:

$$COV(x,y) = \sum (xi - \bar{X}) * (yi - \bar{Y})$$

Equação 2.5. Cálculo da covariância.

Onde:

x_i = valor da variável *X*

X = média da variável *X*

y_i = valor da variável *Y*

Y = média da variável *Y*

Propriedades da Covariância:

CV1. Se *X* e *Y* são independentes, então $Cov(X,Y) = 0$.

CV2. $Cov(a+b\,X\,,\,c+dY\,) = bd.Cov(X,Y\,)$.

CV3. $Cov(X,Y) = E\,(X,Y) - E(X)E(Y)$.

CV4.*Var*(*X*+*Y*) =*Var*(*X*) +*Var*(*Y*) + 2*Cov*.(*X*,*Y*).

CV5.*Cov*(*X*+*Y*,*Z*) =*Cov*(*X*,*Z*) +*Cov*(*Y*,*Z*).

CV6.*Cov*(*X*,*X*) =*Var*(*X*).

CV7.*Cov*(*X*,*Y*) =*Cov*(*Y*,*X*).

Assim definido a covariância e variância, inicia-se a explicação de matriz de covariância que tem em sua forma:

$$MCOV_Y = \begin{pmatrix} \sigma_1^2 & cov(y1y2) & \cdots & cov(y1yn) \\ cov(y1y2) & \sigma_2^2 & \cdots & cov(y2yn) \\ \vdots & \vdots & \ddots & \vdots \\ cov(y1yn) & cov(y2yn) & \cdots & \sigma_n^2 \end{pmatrix}$$

Equação 2.6. Matriz de covariância.

Para calcular a matriz de covariância se calcula covariâncias das variáveis. Exemplo covariância de *x* e *y* tem-se uma matriz:

$$M = \begin{bmatrix} cov(x,x) & cov(x,y) \\ cov(y,x) & cov(y,y) \end{bmatrix}$$

Já que com a propriedade de covariância que diz que *Cov*(*x*,*x*) = *Var*(*x*), então com isso podemos também escrever na forma

$$M = \begin{bmatrix} var(x) & cov(x,y) \\ cov(y,x) & var(y) \end{bmatrix}$$

2.4.4 Correlação

Segundo o dicionário, o termo correlação significa *"relação mútua entre dois termos"*. Já na estatística, é usada para constituir a força da relação entre dois conjuntos de valores.

Mas explicando de modo geral, a correlação é uma medida na qual é obtida a partir da covariância. Essa covariância é dividida pelos desvios padrões de duas variáveis, x e de y, caso sejam maiores que zero.

Segundo o dicionário, o termo correlação significa *"relação mútua entre dois termos"*. Na estatística é usada para constituir a força que dois conjuntos de valores se mantém relacionados.

Pela correlação pode-se verificar o grau de relação entre variáveis, sendo mais próximos do valor um (positivo e negativo) se tiveram um grande grau de inter-relação ou zero caso forem independentes (SOUZA, 2001).

$r_{xy} = -1$ ⟶ Correlação perfeita negativa

$r_{xy} = 0$ ⟶ Correlação nula

$r_{xy} = 1$ ⟶ Correlação perfeita positiva

$-1 < r_{xy} < 0$ ⟶ Correlação negativa

$0 < r_{xy} < 1$ ⟶ Correlação positiva

Os valores dessas variáveis podem ser representados em um diagrama cartesiano, denominado de *diagrama de dispersão*. Nele é possível observar as instâncias dessas variáveis e como elas estão relacionadas.

Para medir o grau de correlação e o sinal dela, a associação linear que existe entre X e Y é dada pela fórmula:

$$Cov(X,Y) = \frac{1}{n}[\Sigma x.y - \frac{\Sigma x.\Sigma y}{n}]$$

Equação 2.7. Cálculo de correlação.

A medida de correlação pode ser feita a partir do coeficiente de correlação linear de Pearson, também conhecido como *coeficiente de correlação produto-momento*, determina o grau de correlação entre duas variáveis de escala intervalar.

$$r_{xy} = \frac{Cov(x,y)}{\sqrt{\sigma_x \sigma_y}} = \frac{S_{xy}}{\sqrt{S_{xx}.S_{yy}}}$$

Equação 2.8. Cálculo da correlação usando coeficiente de Pearson.

As somas de quadrados são calculados da seguinte forma:

$$S_{xy} = \sum x.y - \frac{\sum x.\sum y}{n}$$

Sendo a soma dos quadrados de X e Y, sendo n o número de observações:

$$S_{xx} = \sum x^2 \frac{(\sum x)^2}{n}$$

$$S_{yy} = \sum y^2 \frac{(\sum y)^2}{n}$$

2.5 Características Monoescala

2.5.1 Perímetro

Perímetro é a medida do comprimento, de um contorno. Para o calculo basta efetuar a soma de todos os lados.

Exemplo: O cálculo do perímetro de um retângulo de base *b* e altura h pode ser feito através da fórmula

$$P = b + h + b + h$$

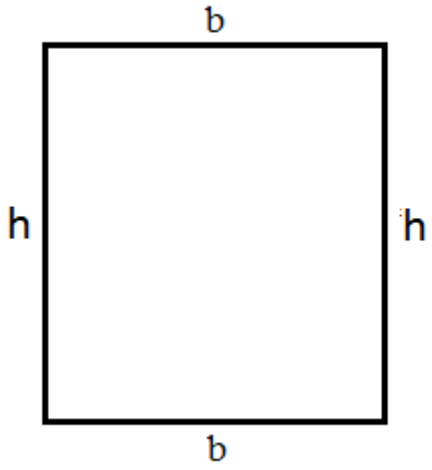


Figura 2.7. Perímetro de um polígono

Colocando a fórmula de outra forma:

$$P = 2(b + h)$$

Equação 2.9. Fórmula matemática do perímetro.

2.5.2 Área

A definição de área se resume em quantidade da superfície do elemento, onde existem varias unidades de medida, entretanto a mais utilizada é o metro quadrado (m^2) e seus múltiplos. A área de um elemento é a somatória da superfície, para formas mais simples como quadrado, retângulo, triangulo, e outros existem cálculos por formulas.

Exemplo: Área de um retângulo de base *b* e altura *h*, o calculo se define como:

$$A = b * h$$

Equação 2.10. Fórmula da área.

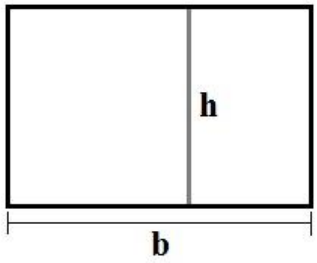


Figura 2.8. Área de um retângulo.

2.5.3 Compacidade

Sejam P e A perímetro e área interna a um contorno, a razão A/P^2 mede a compacidade do contorno.

Compacidade, de acordo com o dicionário, é citada como:

1. Qualidade ou estado daquilo que é compacto, denso.

2. Relação entre o volume aparente de um sólido e o espaço real ocupado por suas partículas. Tal relação varia de acordo com a dimensão dos poros existentes entre as partículas.

2.5.4 Centroide

O centroide é o ponto no interior de uma forma geométrica que define o seu centro geométrico. De acordo com o dicionário:

1. Física - o centro de massa de um corpo.

2. Geometria - numa figura geométrica plana ou tridimensional, ponto que coincide com o centro de massa de um corpo perfeitamente correspondente à figura (isto é, um corpo formado por uma finíssima camada homogênea de material, a recobrir por igual à referida figura). Matematicamente, as coordenadas do centroide são as médias das coordenadas correspondentes de todos os pontos da figura.

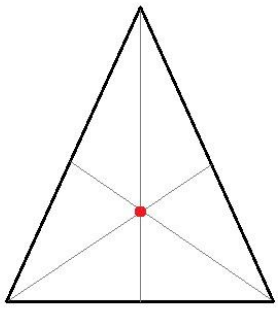


Figura 2.9. Centroide de um triângulo

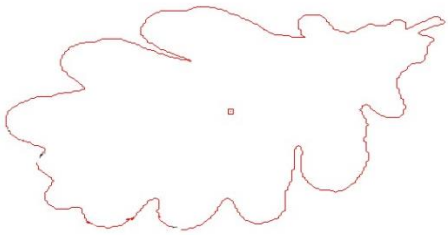


Figura 2.10. Centroide de uma folha

2.5.5 Distância entre dois Pontos

O cálculo da distância entre dois pontos é obtido pela formula:

$$d_{AB} = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$$

Essa formula é facilmente explicada se visualizar esses dois pontos quaisquer *A* e *B* que serão calculados, em um plano cartesiano pode se visualizar um triangulo retângulo, que genericamente pode ser representado por essa figura:

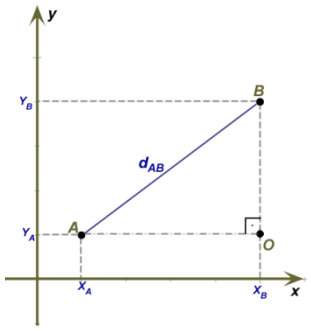


Figura 2.11. Gráfico de representação dos pontos de um triângulo retângulo.

Entretanto essa distancia pode ser obtida pelo teorema de Pitágoras, que diz que a hipotenusa ao quadrado é igual a soma dos catetos ao quadrado, onde a distância equivale a hipotenusa então:

$$d_{AB}^2 = AO^2 + BO^2$$

Entretanto, considerados:

$$AO = x_B - x_A \text{e } BO = y_B - y_A$$

Portanto, a expressão fica da seguinte forma:

$$d_{AB}^2 = (x_B - x_A)^2 + (y_B - y_A)^2$$

E por fim:

$$d_{AB} = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$$

Equação 2.11. Fórmula do cálculo da distância entre dois pontos.

2.5.6 Diâmetro

O diâmetro de uma forma é a maior distancia entre dois pontos pertencentes ao contorno (perímetro).

2.5.7 Maior e Menor Eixos

Maior eixo é onde a forma é mais alongada e perpendicular a ela temos o eixo menor, esses dois eixos são chamados de eixos principais, para o calculo desses eixos precisa-se do contorno dessa forma com isso é calculado os autovetores da matriz de covariância da forma em questão, o autovetor associado ao maior autovalor é o maior eixo da forma, e o segundo maior autovalor esta relacionado ao menor eixo.

2.5.8 Simetria

A simetria pode ser descrita como paridade da relação entre comprimento, altura e largura, ou seja, é a semelhança da forma em volta de um eixo determinado, podendo este ser um ponto, reta ou plano.

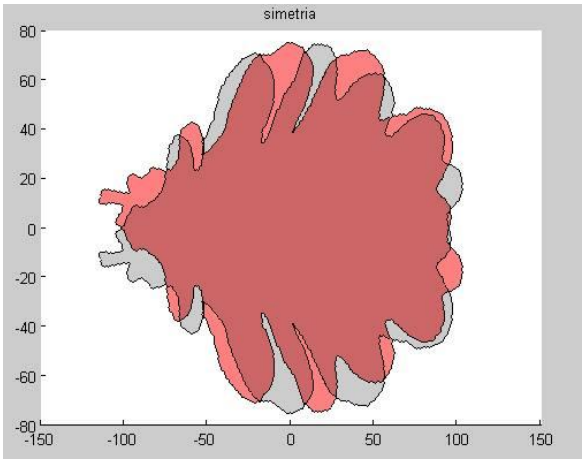


Figura 2.12. Simetria de uma folha.

2.5.9 Curvatura

Seja uma curva sem pontos duplos e com uma tangente determinada em cada ponto.Traçando as tangentes à curva em dois pontos A e B e designemos por a o ângulo formado por estas tangentes. Isto é o ponto de rotação da tangente quando se passa do ponto A para o ponto B. Este ângulo é chamado de ângulo de contingência.Quanto mais inclinada a curva, maior este ângulo.

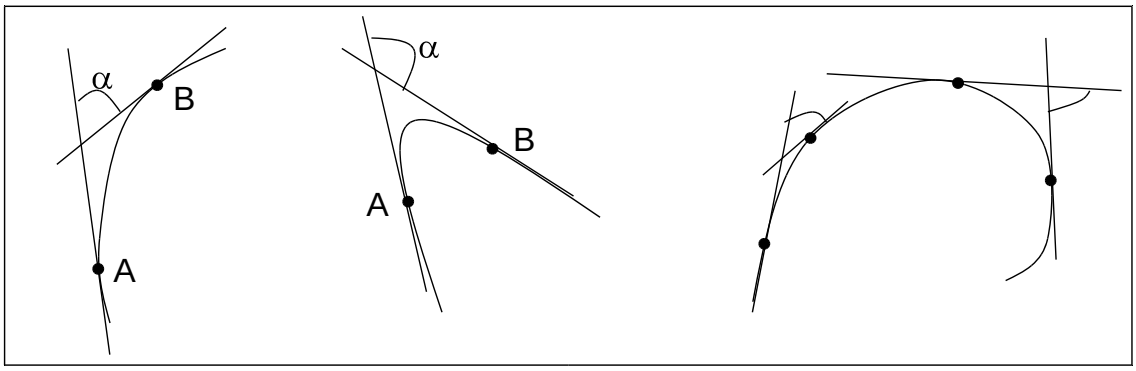


Figura 2.13a. Exemplos de curvas.

Como existem curvas de comprimentos diferentes, caracterizar o grau de encurvamento dos arcos de curva somente pelo ângulo de contingência, não é uma boa ideia. A característica completa de uma curva será então o quociente do ângulo de contingência pelo comprimento do arco correspondente.

A curvatura média Km do arco AB ao quociente do ângulo de contingência correspondente a e do comprimento de arco que ele subentende:

$$K_m = \frac{\alpha}{AB}$$

Considerando uma imagem de dada curva C, para se obter o valor de quanto C se dobra, ou seja, a taxa de mudança de direção das arestas de sua trajetória é necessário o cálculo da curvatura (k).

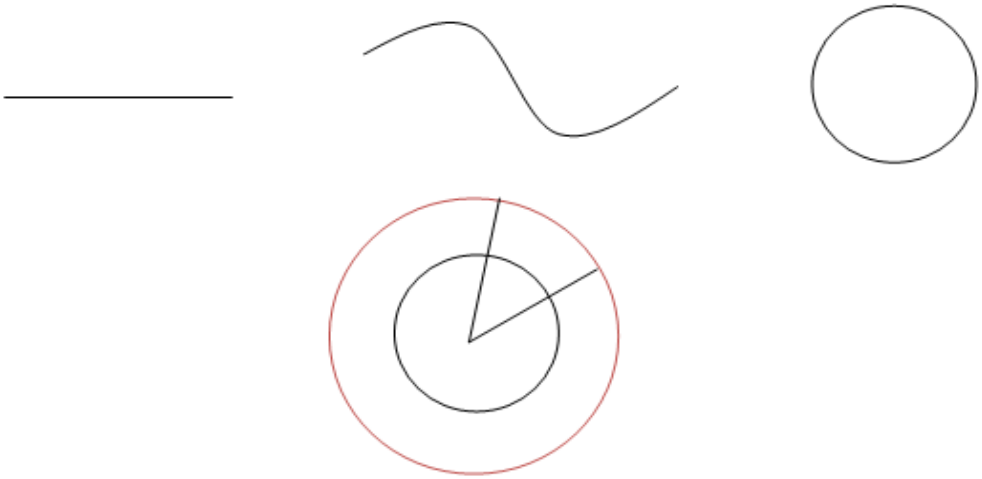


Figura 2.13b. Exemplos de curvatura

Na figura acima, embora o arco seja o mesmo, a curvatura do arco vermelho é menor, que a curvatura do arco preto. As retas são denominadas como curvas simples, que por definição, possuem curvatura igual a zero em todos os pontos.

Dentre o conjunto de curvas, além das retas com k=0, consideram-se os círculos, ou seja, as circunferências. Tendo r como raio do círculo, a curvatura é definida como

$$k = \frac{1}{r}$$

Sendo esta fórmula acima em todos os seus pontos. Portanto, quanto maior o raio da circunferência, menor será sua curvatura.

r (cm)	k
0,5	2
1	1
2	0,5
4	0,25
8	0,125

Pela definição, dada uma curva paramétrica c(t) = (x(t),y(t)), descrita como uma simulação entre o contorno em função do tempo (NICOLIELLO; Heitor, 2007, p.16), podemos calcular sua curvatura com a seguinte fórmula:

$$k(t) = \frac{(x'(t)y^n(t) - xn(t)y^2(t))}{(x'(t)^2 + y'(t)^2)^{\frac{3}{2}}}$$

Equação 2.12 Cálculo da Curvatura.

2.5.10 Momentos

O processamento de imagens como dito anteriormente consiste de vários processos, alguns deles envolvendo a fase de extração de dados como a técnica denominada momentos, introduzida a seguir.

Momentos, ou momentos estatísticos, é um dentre tantos outros métodos usados para o colhimento de dados, mas especificamente características, de uma dada imagem. Estes momentos são calculados através de funções derivadas da imagem que já passou pelo processo de segmentação, expondo a distribuição espacial dos pontos inclusos na imagem.

Podemos definir os momentos regulares de uma imagem a partir da fórmula:

$$m_{pq} = \sum_{1}^{nx}\sum_{1}^{ny} x^p y^q f(x,y)$$

Equação 2.13. Cálculo de Momentos Regulares.

A variável *mpq*representa o momento (de ordem $p+q$) que queremos calcular; *nx* representa a largura e *ny* a altura da imagem; *f(x,y)* é a função de intensidade da imagem binária, portanto só contendo os valores 0 e 1.

Ao se calcular o momento de ordem 1, o valor obtido será o mesmo da média de uma variável. Essa média entra no conjunto de características da variável analisada, sendo ela uma propriedade de caráter estatístico invariante.

Podemos descrever o momento de ordem 1 de uma variável como:

$$m_{pq} = \frac{1}{N}\sum_{i=1}^{l}\sum_{j=1}^{c} x_{ij}$$

Equação 2.14. Cálculo de Momentos de Ordem 1.

A média, nada mais é do que, a soma de todos os elementos da coluna j contidos na linha *i*, divididos pelo número de elementos somados, representados por N. As variáveis *l* e *c* representam o numero de linhas e colunas, respectivamente.

Já o momento de ordem 2 equivale ao calculo da variância de uma variável, onde o valor contido na linha *i* e coluna *j* de uma matriz com *N*-1 elementos dessa linha é subtraído pelo valor da média obtida elevada a ordem, neste caso, a ordem = 2, podendo ser representada por:

$$m_{pq} = \frac{1}{(N-1)} \sum_{i=1}^{l} \sum_{j=1}^{c} (x - \bar{X})^2$$

Equação 2.15. Cálculo de Momentos de Ordem 1.

Com isso, podemos deduzir que o cálculo do momento de ordem *n* pode ser colocado na seguinte fórmula

$$m_{pq} = \frac{1}{(N-1)} \sum_{i=1}^{l} \sum_{j=1}^{c} (x - \bar{X})^n$$

Equação 2.16. Cálculo de Momentos de Ordem N.

calculando momentos de acordo com a ordem dada.

Dessa forma, ao identificar momentos que ocorrem regularmente, medidas podem ser definidas em relação ao objeto analisado, como o calculo do centroide ou baricentro.

2.5.11 Transformada de Fourier

A transformada de Fourier é uma importante ferramenta utilizada no processamento de imagens digitais.

Representando uma imagem em 2D no espaço, considerando-a como um sinal também em duas dimensões, utilizando a transformada de Fourier, podemos analisa-la como sinusóides espaciais.

Através da transformada de Fourier, é possível analisar uma imagem como um conjunto de sinusóides espaciais em várias direções, tendo cada sinusóide uma frequência precisa (Faria, 2010).

A transformada de Fourier de uma imagem u(t) é dada por

$$U(f) = F\big(u(t)\big) = \sum_{-\infty}^{+\infty} u(t)e^{-i2pift}dt$$

Equação 2.17. Transformada de Fourier de um sinal.

e sua inversa é denotada por

$$u(t) = F^{-1}\big(U(f)\big) = \sum_{-\infty}^{+\infty} U(f)e^{i2pift}df$$

Equação 2.18. Transformada de Fourier e sua inversa.

2.6 ESTATÍSTICA MULTIVARIADA

Atualmente existem numerosos métodos envolvendo a análise de múltiplas variáveis, com finalidades diversas.

A Estatística Multivariada tem sido uma ótima ferramenta para o estudo e tratamento de várias variáveis de forma simultânea. Pode ser aplicada aos dados mesmo quando não se tem definida a ideia do modelo teórico envolvendo o relacionamento entre as variáveis.

Entretanto,antes de escolhermos um método para trabalhar, é necessário saber qual se adequa aos dados. Para o relacionamento entre amostras de dados existem métodos favoráveis para se trabalhar, assim como para previsão de resultados. Testá-los, verificando qual se aqueda a cada situação é fundamental.

Continuando com o estudo da estatística multivariada, a seguir falaremos sobre a análise de componentes principais e a análise de discriminante.

2.6.1 PCA

Esta seção conterá algumas informações elementares, conhecimentos matemáticos que serão necessários para compreender o processo de Análise de Componentes Principais que baseiam-se em covariância, autovetores, autovalores, componentes essenciais para a cálculo da PCA.

2.6.1.1 O Processo da PCA

A análise de componentes principais (PCA) foi introduzida por Karl Pearson em 1901, porém o tratamento formal do método é devido ao trabalho de Hotteling, na década de 30, segundo (VASCONCELOS,2014).

A PCA é um método que tem por finalidade básica, a obtenção de um pequeno número de componentes principais de um conjunto de variáveis, que retenham o máximo possível da informação contida nas variáveis originais, simplificando então o processo de visualização e classificação de dados. Este método tem como saída um conjunto novo de variáveis, denominadas *componentes principais*. Os componentes principais são ortogonais entre si e nada mais são que a combinação linear das variáveis usadas originalmente.

Os componentes são ordenados na ordem de maior importância (retém o máximo de informações) para o de menor importância, tendo esses componentes então uma maior contribuição.

O principal efeito é o alinhamento do eixo principal dos dados com o maior autovalor encontrado em um novo sistema de coordenadas cuja origem é o centroide do conjunto de dadoscomo demonstrado na figura 2.14.

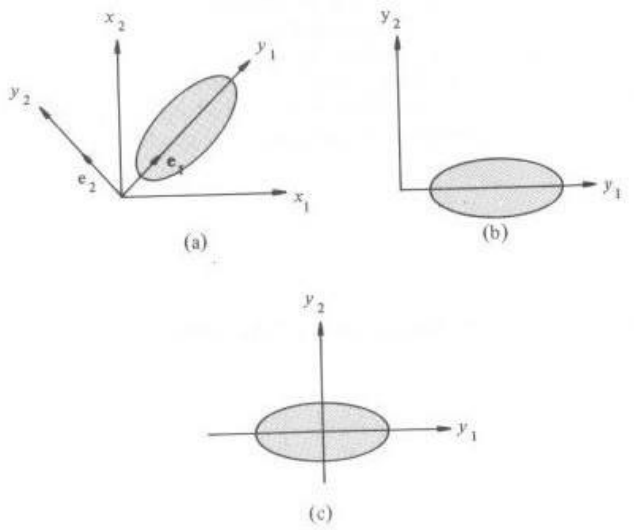


Figura 2.14. Rotação de uma figura em um espaço 2D

O primeiro componente principal pode ser descrito como um eixo único no espaço, onde projetando informações nesse eixo os resultados formados a partir dos valores resultarão em uma nova variável.

Perpendicular ao primeiro componente principal se encontra o segundo componente principal. Da mesma forma que o primeiro, ao projetar informações no eixo, uma nova variável será formada.

A análise das componentes principais é, portanto, uma técnica de transformação de variáveis, este método graficamente pode ser descrito como a rotação de pontos existentes num espaço multidimensional originando eixos.

Entretanto, podemos utilizar a PCA para finalidades diferentes. Quando não retiramos dimensões dela, ou seja, utilizando todos os seus componentes, no qual o objetivo seja descorrelacionar eixos ou alinhar uma figura, fazendo uso de seus maior e menores eixos, denominamos como Transformada Karhunen-Loève (TKL). Já a PCA seleciona os componentes principais responsáveis pela maior parte da variância, para então criar uma nova matriz de dados transformados. E mesmo tendo como meta facilitar a visualização dos múltiplos dados, o conjunto dos componentes principais é tão grande quanto o conjunto composto pelos dados originais analisados.

2.6.1.2 Sequência de cálculos da PCA

Os passos para calcular as componentes principais são:

- Obter os dados; calcular a média destes dados;
- Subtrair a média de todos os itens de dados;
- Calcular a matriz de covariância;
- Calcular os autovalores e autovetores da matriz de covariância;

Passo 1: Obtenção dos dados

O conjunto de dados analisados dá-se pelo o vetor de coordenadas (x,y) do perímetro da figura.

Passo 2: Subtrair à média

É subtraído a média dos dados, para que os novos dados gerados sejam os dados ajustados na origem para que a PCA seja executada corretamente.

Passo 3: Calcule a matriz de covariância

Esse passo é feito como o demonstrado *na sessão 2.3.2* onde os dados são de dimensão = 2, gerando:

$$MCOV = \begin{bmatrix} var(x) & cov(x,y) \\ cov(y,x) & var(y) \end{bmatrix}$$

Passo 4: Calcular os autovetores e autovalores da covariância matriz

São tirados autovalores e autovetores dessa matriz de covariância, a seguir vão ser ordenados os valores do autovalor, para que obtenha uma matriz cujo autovetores corresponde ao autovalores, assim nessa ordem os autovalores tem mais facilidade de identificar os autovetores cujos autovetores tem pouca informação, após ordenados os autovetores vão ser transposto e multiplicado pelos dados ajustados para que seja rotacionado.

2.6.2 Análise de Discriminante

A análise discriminante consiste de uma técnica da estatística multivariada, que tem por finalidade classificar e discriminar objetos. É a técnica da estatística multivariada que estuda a separação de objetos de uma população em duas ou mais classes (KHATTREE & NAIK, 2000).

Na análise dediscriminante, o grupo de dados estudado já passou pelo processo de análise de componentes principais, portanto ela trabalha com informações de agrupamentos conhecidos.

Neste processo, é suposto que corretamente estão classificadas as observações, logo, cabe a análise discriminante verificar se estão discriminados os grupos, buscar e classificar observações que sejam desconhecidas e ver quais variáveis são mais importantes.

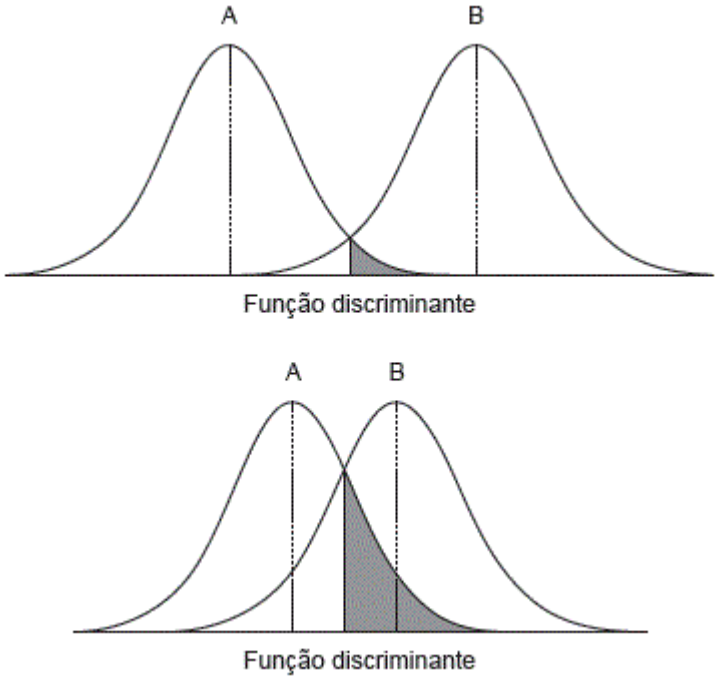


Figura 2.15. Representação univariada de escores Z discriminantes.
Fonte: Hair et al. (2005)

2.6.3 Análise de Discriminante linear

A análise de discriminante linear tem por finalidade decidir em qual de dois grupos um determinado indivíduo pertence, sendo vantajosa no quesito de ter fácil visualização em termos geométricos e poder substituir um conjunto original das medidas por um valor único. Esse valor será definido como uma combinação linear.

Na análise de discriminante linear, é possível discriminar duas ou mais classes sem grande complexidade, procurando combinações lineares(X_a) das variáveis observadas do estudo(p) que obtenham um melhor resultado de separá-las em designados subgrupos de indivíduos.

Os resultados X_a adquiridos constituem-se de eixos discriminantes ou variáveis canônicas. Pode ter sua utilização para ter representações gráficas que evidenciem a diferenciação entre classes, sendo útil também para a classificação de indivíduos porvir.

2.7 Mineração de dados

Atualmente, a mineração de dados tem atraído cada vez mais o mundo da tecnologia em geral. Através de métodos utilizando transformações de grandes quantidades de dados em regras e padrões, facilitou a forma de realizar seleções de dados pertinentes, abrangendo milhares de técnicas de transformação, observação e exame de dados.

A mineração de dados pode ser dividida em mineração direcionada e não-direcionada. Na direcionada tenta-se prever particularidades de dados em um ponto. Como exemplo, podemos citar o preço estimado de um veículo popular à venda, baseado em dados colhidos sobre outros veículos. Já na mineração de dados não direcionados, busca-se criar grupo de dados, ou padrões existentes sobre eles.

A finalidade da mineração de dados é a criação de um modelo que melhore de alguma forma o entendimento e discriminação dos dados obtidos e os que possam a ser extraídos.

2.7.1 ModeloIBK

O classificador IBK é um modelo com uma construção simples baseado em instâncias, envolvendo distâncias e vizinhança (K-vizinhos). Esse algoritmo utiliza a ideia de que em um conjunto de treinamento, as instâncias utilizadas formam um modelo para os dados. Para examinar uma instância nova do conjunto de teste, calcula-se levando em consideração as instâncias memorizadas anteriormente, o intervalo entre a distância de entrada e a distância mínima.

O IBK permite que ajustes sejam feitos no algoritmo, como por exemplo o parâmetro referente ao número de vizinhos mais próximos (N) e a função da distância entre duas instâncias (d). Utilizando o algoritmo sem modificações, deixando por padrão N = 1 e d(x,y), observa-se que o IBK calcula o resultado usando a fórmula da distância e entre dois pontos, já citada anteriormente na seção 2.5.5.

2.7.2 Modelo NaiveBayes

Um classificador Bayes é um classificador bayesiano probabilístico simples que utiliza o Teorema de Bayes na criação de um modelo de dados.

Explicando de maneira simples, um classificador que faz uso do algoritmo NaiveBayes interpreta a presença ou a ausência de uma dada classe, mostrando se está ou não ligada a outro artifício. Dando como exemplo, uma folha pode ser considerada como de pitanga se possui um determinado tamanho, área e simetria. Esses atributos, dependentes ou não, o algoritmo de NaiveBayes considera todas as particularidades para que assim contribuam de forma independente para a chance de que a folha estudada seja a da pitanga.

Podemos explicar teoricamente que no modelo NaiveBayes, dada um classe (representada com a letra c) e um conjunto de valores (x_1,x_2,...,x_n), o algoritmo faz uma aproximação da probabilidade P (C = c | X = x1...Xn = xn) em conjunto de dados com probabilidades já conhecidas, por sucessivas decomposições. Proporções são usadas em atributos de enumeração para probabilidades e em distribuição normal para numéricos (utilizando a média e o desvio padrão em instâncias do conjunto de teste).

2.7.3 Modelo de Árvore de decisão

Os modelos de árvore de regressão logística linear e indução, que combinam modelos lineares e e indução de árvores de decisão, têm sido bastante populares para a resolução de problemas relacionados a classificação de dados.

O modelo de árvore de logística é uma representação da estrutura de uma tabela de decisão sob a forma de uma árvore. Esta árvore é construída atráves de funções de regressão logísticas em seus nós-folha. Dentre os algoritmos presentes neste grupo podemos citar o algoritmo de árvore de decisão J48 e o LMT.

2.7.3.1 J48

O J48 é um algoritmo de árvore de decisão baseado em instâncias do conjunto de teste, utilizado para classificá-las, agrupando-as recursivamente, minimizando a variação das classes em cada subgrupo. Gerando árvores de decisão, o J48 estima em cada nó a importância ou existência de cada atributo, utilizando os valores de atributos do conjunto de teste em não nós-folha e as instâncias em nós-folha. Isso faz com que cada nó no interior da árvore seja uma decisão de classificação.

2.7.3.2 LMT

O modelo LogisticModelTrees (LMT) trabalha de forma a unir modelos que envolvem a regressão logística com árvores de indução, utilizado para resolver tópicos que envolvam a classificação de um grupo de dados. Possui a vantagem de poder estimar a probabilidade de todas as classes envolvidas.

Este modelo representa os nós dependendo do tipo de atributo existente. Para atributos do tipo numérico, a divisão é binária, mas sendo categórico, um ramo é criado para os possíveis valores que o atributo possuam vir a ter. Caso o atributo contenha valores desconhecidos, calcula-se do grupo de treinamento informações adicionais, como a média (atributos numéricos) e moda (atributos categóricos), preenchendo então os valores antes desconhecidos pelos calculados.

Capítulo 3
DESENVOLVIMENTO

Neste capítulo, etapas serão apresentadas para a realização do projeto proposto inicialmente.

Primeiramente será realizada a especificação de requisitos do projeto, descritos na seção 3.1. Na seção 3.2 a implementação do sistema é apresentada. Na seção seguinte serão discutidos os resultados obtidos.

3.1 REQUISITOS PRINCIPAIS DO PROJETO

O projeto proposto de classificação de folhas usando medidas invariantes deverá:

a) ler um arquivo de imagem digital de folhas;

b) transformação da imagem colorida em escala em cinza, aplicando posteriormente a binarização da imagem, retirando ruídos do objeto de interesse;

c) utilizar meios de extração de dados obtidos a partir da análise das imagens das folhas;

d) implementar o algoritmo utilizando o MATLAB®;

e) apresentar os resultados obtidos;

3.2 IMPLEMENTAÇÃO

Nesta subseção será descrito o processo da implementação e as técnicas e ferramentas utilizadas.

3.2.1 Técnicas e ferramentas utilizadas

A implementação do algoritmo foi desenvolvida utilizando a linguagem de programação do software matemático pago MATLAB®, utilizando a versão R2009a. Para a comparação de dados e das características obtidas na etapa de extração de dados, foi utilizado o pacote de software de distribuição livre Weka (WaikatoEnvironment for KnowledgeAnalysis), criado em 1993 na Universidade de Waikato, Nova Zelândia.

3.2.2 Implementação do sistema

A seguir, fragmentos do código serão apresentados e a descrição das etapas do desenvolvimento do algoritmo.

Para o desenvolvimento do projeto em questão, foram utilizadas funções contidas na biblioteca do MATLAB®.

3.2.2.1 Obtenção da imagem

Para a classificação proposta inicialmente é necessária à obtenção das imagens como base para a análise. No caso deste projeto em questão, foram disponibilizadas imagens do banco de dados de folhas da Mata Atlântica pelo Professor Orientador deste projeto, contendo duzentas e quarenta e oito folhas de dez espécies, sendo cinco delas conhecidas (coque-casa, jequitibá, lima, limão e pitanga) e cinco desconhecidas.

3.2.2.2 Obtenção da imagem utilizando arquivo

A realização da leitura da imagem a partir de arquivo é feita utilizando o método `imread(filename,fmt).`

`A = imread(filename,fmt)` lê uma imagem em L tons de cinza ou colorida do arquivo especificado pelo nome do arquivo. Caso o arquivo não se encontre na pasta atual, ou em uma pasta no caminho MATLAB®, é necessária a especificação completa do mesmo. A string`fmt` corresponde ao tipo de extensão utilizada pelo arquivo.

3.2.3 Segmentação

Esta seção refere-se ao processo de dividir a imagem em segmentos (regiões) a fim de simplificar a representação da mesma.

3.2.3.1 Algoritmo de Binarização

O algoritmo de binarização de imagens compreende a etapa de segmentação de imagens, gerando uma imagem binária através da técnica chamada de Limiarização (“*Thresholding*”), atribuindo o valor 0 a intensidade menores que *T* e um para os demais pixels constituintes da imagem. A saída deste algoritmo será uma nova imagem de atributo lógico.

Antes de realizar de fato a binarização da imagem, primeiramente foi feita a conversão da mesma, de tipo RGB para escala em cinza pela função `rgb2gray(I)` recebida pela variável *Y*. Essa função elimina a tonalidade e saturação da imagem, mas mantendo a informação de luminância (b).

Quadro 3.1 – Função de binarização

```
1   function B = FBinarizacao(I)
2 -     Y = rgb2gray(I);
3 -     Y = 255-Y;
4 -   T = graythresh(Y);
5 -   B = im2bw(Y,T);
6 -   end
```

Na linha 3, é feita a inversão de valores, a fim de um melhor resultado com imagens que possuam fundo branco ou claro (c). A linha 4 é responsável pelo processo de cálculo de um limiar que será usado para a conversão da imagem de intensidade em tom de cinza para outra binária pelo método de Otsu. A partir do valor do limiar retido por T, finalmente é feita a binarização pelo comando `im2bw(imagem_em_escala_de_cinza, limiar)`, podendo ser observado como consequência na figura abaixo (d).

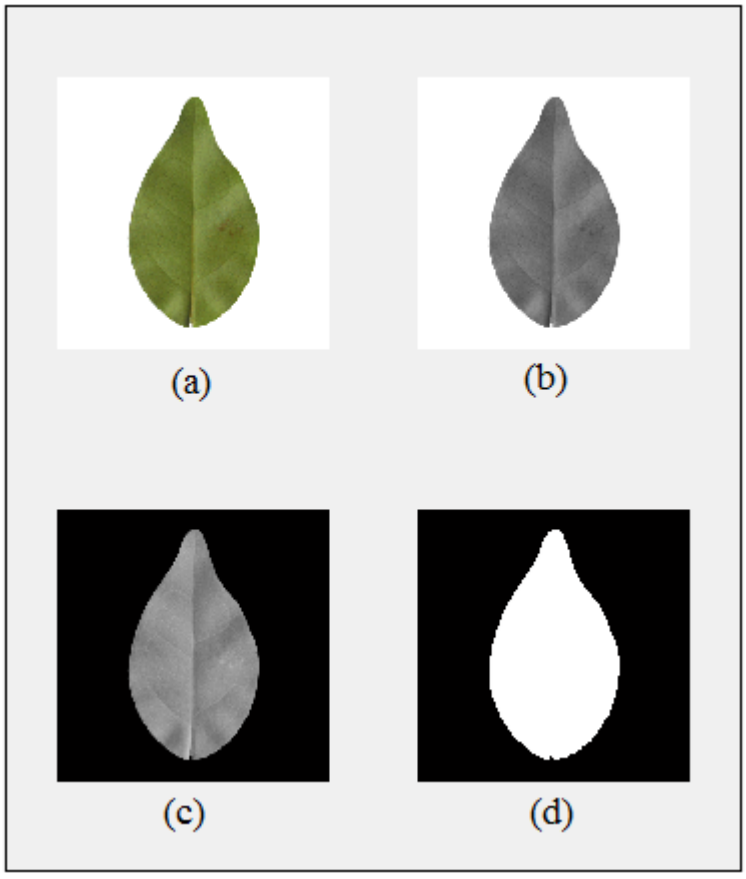


Figura 3.1. Processo de Binarização da folha

3.2.3.2 Algoritmo de Limpeza de Ruídos e cálculo da Área

O cálculo da área na imagem foi obtida pela quantidade de pixels no elemento analisado, no caso deste estudo, a folha.

Para se iniciar o cálculo da área é necessário obter a região, onde para se estimar área da folha, é somada a quantidade de pixels. O método é descrito no código abaixo, realizando o tratamento da imagem com a função “LimpezaRuidos”, calculando a área após a localização do maior objeto na figura. Na linha 17, a função `find(L == maior)` devolve como resultado todas as coordenadas associadas a aquele objeto, somando somente a quantidade de coordenadas que foram retornadas.

Quadro 3.2 – Função para limpeza de ruídos

```
1  function [A,AreaFolha] = LimpezaRuidos(B)
2  [L,num] = bwlabel(B,8);
3  maiorid = 0;
4  somav = 0;
5
6  [a,b] = size(L);
7
8  for i=1:1:num,
9    [l,c,ve] = find(L==i);
10   soma = sum(ve(:));
11   if soma > somav,
12       somav = soma ;
13       maiorid = i ;
14   end
15 end
16
17 [l,c,ve] = find(L==maiorid);
18 AreaFolha=sum(ve(:));
19
20 A=L;
21 for i=1:1:a,
22   for j=1:1:b,
23     if L(i,j) ~= maiorid
24       A(i,j)= 0;
25     end
26   end
27 end
28 end
```

A área só é calculada na maior região encontrada, pois a limpeza de ruídos elimina todas as outras regiões que não sejam da folha, para que alguns ruídos não interfiram nos cálculos seguintes.

Esse cálculo retorna a área obtida em pixels.

3.2.4 Extração de dados

Nesta sessão serão apresentados algoritmos relacionados a cálculos para à extração de dados da folha, através de funções aritméticas e cálculos algébricos.

3.2.5 Algoritmo para Encontrar Contorno (Método do ceguinho)

Iniciando a etapa de extração de dados, a obtenção do contorno da imagem folha é uma operação trivial. Através dela, poderemos extrair o contorno da imagem e obter outros dados, como a extração particular dos pixels da borda da folha e posteriormente calcular a curvatura da mesma.

O algoritmo da função de contorno recebe a imagem que passou pelo processo de segmentação, guardando as dimensões em uma matriz 2x2, compreendendo ao número de linhas e colunas e então procura pelo primeiro ponto do contorno com a função `find(img')`. A posição inicial encontrada é representada como um sinal complexo. Ao encontrar este primeiro ponto, os pixels vizinhos são identificados retornam então a localização de cada um deles na cadeia. Encontrando um outro ponto compatível com a borda da folha, logo após será feito o cálculo do segundo pixel da borda e a direção em que se encontra, fazendo novamente o processo de identificação dos pixels vizinhos do ponto encontrado e assim em diante até todos os pontos da borda serem descobertos.

Quadro 3.3. Trecho do código de contorno.

```
while(~bol)
    de = dpc;
    di  = mod(dpc + 1, 8);
    Pe = chainpoint( E(1), de);
    Pi = chainpoint(E(1), di);

    if (~(img(imag(Pe),real(Pe))) && (img(imag(Pi),real(Pi))))
        bol = 1;
    else
        dpc = dpc + 1;
    end
```

3.2.6 Algoritmo para Extração do Contorno

O algoritmo para a extração de contorno recebe o contorno da função `contour` e extrai somente a borda da folha. A partir dessa função também é calculada a curvatura e em

consequência, seu morfograma, obtendo picos de frequência, guardando os dados em uma matriz.

Quadro 3.4. Trecho da função de extração de contorno

```
[kaux rf] = curvogram(E, t);
plot(rf);


for i = 1 : max(size(kaux))
    k(i, t) = kaux(i);
end
```

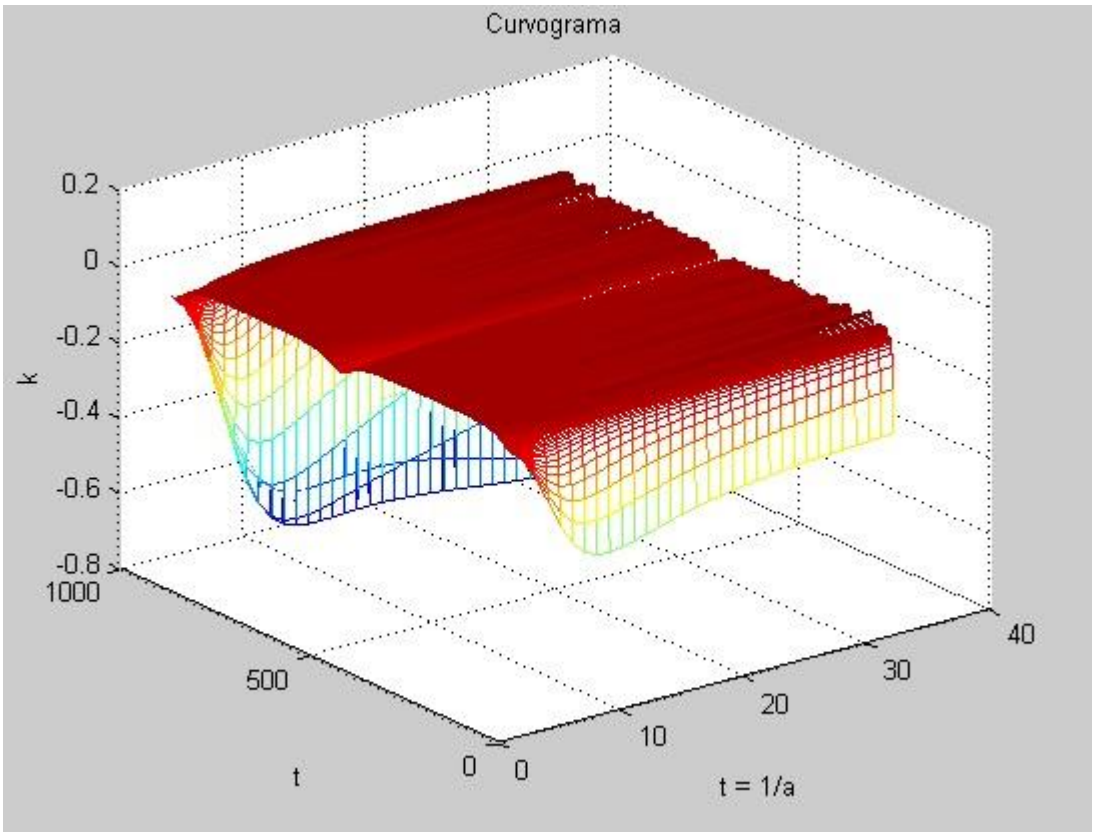


Figura 3.2. Curvograma de uma folha.

O curvograma tem como entrada o vetor calculado anteriormente com coordenadas parametrizadas, sendo estas coordenadas resultado da etapa de extração do contorno da folha. No código, é utilizada a transformada de Fourier utilizando o filtro da Gaussiana suavizando sinal.

3.2.7 Algoritmo da PCA (TKL)

O cálculo da PCA (Principal ComponentAnalysis) tem por objetivo a peneiração dos dados coletados a partir da imagem digital da folha analisada. A saída da função será a variável `FinalDataAdjust`com os dados analisados.

Quadro 3.5a. Trecho de código para o calculo da PCA

```
1   function FinalDataAdjust = PCA(borda)
2   Dados=borda;
3   Media=mean(Dados);
4
5   DataAdjust(:,1)= Dados(:,1) - Media(1);
6   DataAdjust(:,2)= Dados(:,2) - Media(2);
7
8   Mcovariancia=cov(Dados);
9
10  [v,lambda]=eig(Mcovariancia);
11
12  lamb=diag(lambda);
13  [lb,ind]=sort(lamb,1,'descend');
14
15  NewVet=v(:,ind);
16
17  rmax=[Media-200*(NewVet(:,1)).'; Media+200*(NewVet(:,1)).'];
18  rmin=[Media-200*(NewVet(:,2)).'; Media+200*(NewVet(:,2)).'];
19
20  RowFeature=NewVet.';
21
22  RowDataAdjust=DataAdjust.';
23  FinalDataAdjust=(RowFeature*RowDataAdjust).';
```

Na linha 5 a variável *DataAdjust*(:,1) recebe o valor da subtração dos dados correspondente a valores de *x* da primeira linha da matriz contendo os dados obtidos menos a média dos mesmos. Na linha abaixo é feito o mesmo processo, mas com o cálculo dos valores de *y*. Na linha 8 é feito o cálculo da covariância pela função `cov(Dados)`.

A variável `NewVet` na linha 15 recebe os autovalores de acordo com a ordem decrescente dos autovalores, onde cada vetor é um vetor coluna de m linhas por uma coluna. Os vetores são os novos eixos, perpendiculares entre si, daí então é feito o calculo dos eixos para a plotagem no gráfico.

Na linha 20 é realizada a rotação dos dados já ajustados, passando a matriz inversa da variável `NewVet`.

Quadro 3.5b. Continuação do código da PCA

```
figure,
subplot(2,2,1);
plot(Dados(:,1),Dados(:,2),'k.',rmax(:,1),rmax(:,2),rmin(:,1),rmin(:,2),'k-')
title('dados')

rmax=[-200*(NewVet(:,1)).'; 200*(NewVet(:,1)).'];
rmin=[-200*(NewVet(:,2)).'; 200*(NewVet(:,2)).'];
rmax=[-300 0; 300 0];
rmin=[-0 -250; 0 250];

subplot(2,2,2);
plot(DataAdjust(:,1),DataAdjust(:,2),'k.',rmax(:,1),rmax(:,2),rmin(:,1),rmin(:,2),'k-')
title('Dados ajustados (media zero)')

FinalData=((NewVet.')*(Dados.')).';
udat=mean(FinalData);
xdat=[-400 udat(2);200 udat(2)];
ydat=[udat(1) -100;udat(1) -500];

subplot(2,2,3);
plot(FinalData(:,1),FinalData(:,2),'k.',xdat(:,1),xdat(:,2),ydat(:,1),ydat(:,2))
title('DadosRotacionados=([Autovetores]T.[Dados]T)T')

rmax=[-300 0; 250 0];
rmin=[-0 -200; 0 200];

subplot(2,2,4);
plot(FinalDataAdjust(:,1),FinalDataAdjust(:,2),'k.',rmax(:,1),rmax(:,2),rmin(:,1),rmin(:,2))
title('Dados media zero rotacionados pelo maior eixo (grosso)')
end
```

3.3 Outros cálculos realizados

Esta etapa de implementação compreende no processo de extração características monoescala.

3.3.1 Perímetro

O cálculo do perímetro foi obtido com a somatória da quantidade de coordenadas retornadas do processo de contorno obtidos do algoritmo do ceguinho, onde as coordenadas foram armazenadas em forma de vetor de números complexos $z = xi + y$.

Esse somatório resultara em quantidade de pixel do contorno;

3.3.2 Compacidade

A compacidade foi obtida multiplicando o perímetro elevado ao quadro, dividido pela área total da folha analisada.

3.3.3 Diâmetro

O diâmetro da imagem é obtido comparando todas as distâncias de todos os pontos, dois a dois, que pertencem ao perímetro, o código começa com a maior distancia zerada e ao decorrer das comparações essa distancia vai sendo substituída por outra maior até que as comparações de todos os pontos sejam concluídas. Nessa função que calcula o diâmetro tem de saída a distância e as coordenadas da maior distância.

Na linha 7 referente ao trecho de código *(ImB(i,1),ImB(i,2),ImB(j,1),ImB(j,2))*, ImB(i,1) e ImB(i,2) correspondem à coordenada [xi;yi] e ImB(j,1),ImB(j,2) correspondem à coordenada [xj;yj].

Quadro 3.6. Função para obtenção do diâmetro.

```
1   function [] = Diametro(ImB)
2   [n,a] = size(ImB);
3
4   dmax=0;
5   for i=1:1:(n-1),
6       for j=(i+1):1:n,
7           d=DistamciaPontos(ImB(i,1),ImB(i,2),ImB(j,1),ImB(j,2));
8           if(d>dmax)
9               dmax=d;
10              ID(1,1)=ImB(i,1);
11              ID(1,2)=ImB(i,2);
12              ID(2,1)=ImB(j,1);
13              ID(2,2)=ImB(j,2);
14          end
15      end
16  end
17
18  plot(ID(:,1),ID(:,2),'--rs','LineWidth',2)
19  plot(ImB(:,1),ImB(:,2), 'black', 'LineWidth', 2)
20  end
```

3.3.4 Momentos

Englobando a etapa de extração de características, o cálculo de momentos de ordem *n* foi feita a partir da fórmula disponibilizada anteriormente, calculando o valor contido na linha i e j elevado a ordem desejada dividido pelo numero total de elementos -1.

Quadro 3.7. Trecho do código para cálculo de Momentos.

```
10  for i=1:1:l,
11      for j=1:1:c,
12          M(i,:) = Momentos(i) +((Dados(i,j) - Media(i))^n);
13      end
14      M(i,:) = Momentos(i)/(N-1);
15  end
```

Esse trecho de código retorna uma matriz linha com N elementos.

3.3.5 Centroide

Para a obtenção do centroide da imagem, foram utilizados os pontos das bordas mais distantes de cada lado, tanto no sentido horizontal quanto no vertical. A partir disso então é feita a divisão dos pontos obtidos de cada direção, divididos pela área.

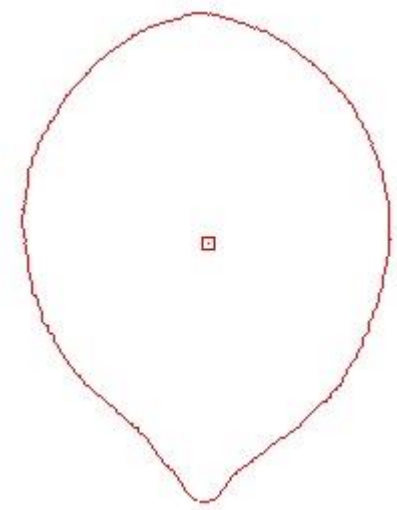


Figura 3.3. Centro da folha de pitanga 001_a

3.3.6 Distância média à borda

A função para o cálculo da distância média até a borda tem como entrada uma função complexa, retornando a distância média, ou seja, ela calcula a distância média da borda até o centro da figura até o centro de massa ou centro igual a zero.

3.3.7 Distância entre Pontos

Seguindo a forma da geometria analítica, o cálculo da distância entre pontos foi feito da seguinte forma:

```
d = sqrt(((x-x1)^2)+((y-y1)^2));
```

A variável *d* retorna o valor da distância calculada pela fórmula acima.

3.3.8 Simetria

A função que calcula a simetria da folha primeiramente compara a folha com sua forma invertida, guardando o número de pixels em comum da intersecção entre elas.

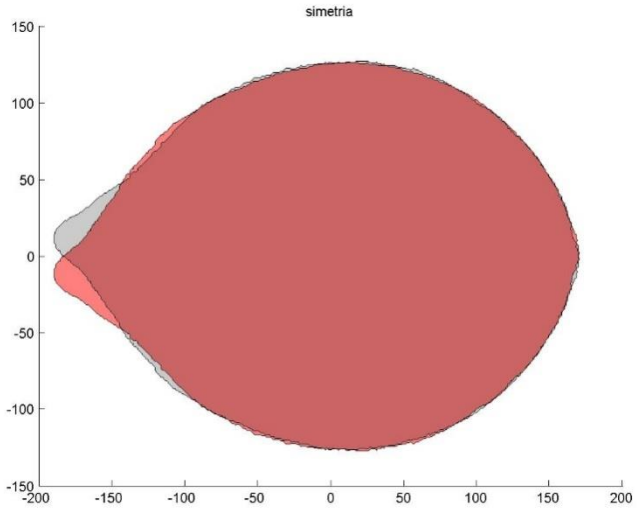


Figura 3.4. Imagem de uma folha de pitanga e sua cópia espelhada.

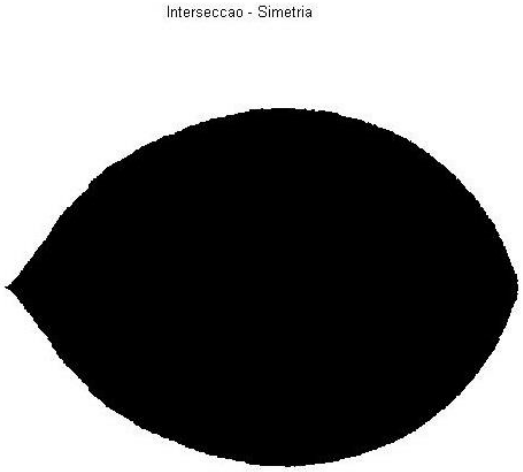


Figura 3.5. Folha simetrizada

3.4 INICIANDO PROCESSO DE CLASSIFICAÇÃO DE FOLHAS

Uma das maiores dificuldades referentes a estatística com múltiplas variáveis é a visualização dos mais diversos dados e poder de alguma forma relacioná-los.

A função *plot* do MATLAB® mostra como saída um gráfico na qual duas variáveis são relacionas. Já o comando *plot3* e *surf* exibem gráficos de visualização 3D. Entretanto, como realizar a visualização de mais variáveis? Esta é a questão que tentamos resolver adiante, procurando maneiras de visualização e relação entre elas, através do cálculo da PCA (Principal ComponentAnalysis).

3.4.1 Análise dos Componentes Principais (PCA)

Para encontrar os componentes principais foi utilizada a função *princomp* disponibilizada pela biblioteca do MATLAB®. É necessário o uso dos dados padronizados para podermos analisá-los corretamente.

3.4.2 Carregando dados

O cálculo dos principais componentes inicia-se através do carregamento dos dados da amostra, composta por onze diferentes variáveis referentes a dados numéricos colhidos das mais de 200 folhas da Mata Atlântica disponibilizadas anteriormente. Estes atributos incluem a área da folha, momentos, perímetro, compacidade, média de picos e valas, pontos do eixo x e y do centroide, distância média à borda e simetria.

Primeiramente é feita a leitura a partir de um arquivo .txt para o MATLAB®. Após feito este procedimento, as colunas formadas apenas por dados numéricos são carregados para uma matriz e os nomes das variáveis para outra.

3.4.3 Calculando Componentes principais

Ao abstrairmos dados das folhas, nem todos os dados possuem uma mesma unidade de medida (cm, m, kg, ml, etc...), portanto, o uso de escala de dados ou outros métodos são preferíveis de utilização.

Considerando os dados como X, para padroniza-los têm-se a seguinte fórmula:

```
media = mean(dados);
desvp = std(dados);

for i=1:11,
       for j=1:248,
                dadosaj(j,i) = (dados(j,i) – media(1,i)/desvp(1,i);
       end
end
```

Listagem 3.1. Trecho da PCA

A média corresponde a uma matriz 1x11 contendo o valor mediano de cada coluna da matriz original de dados, calculado através da função *mean*. O desvio padrão de cada coluna também corresponde a uma matriz 1x11, pela função std do MATLAB®. A variável *dadosaj*é uma nova matriz que receberá os valores dos dados padronizados.

Após o processo de padronização de dados, finalmente é iniciada a análise dos principais componentes. Neste projeto, usaremos a função *princomp*.

```
[coeff,score,latent,tsquared] = princomp(dadosaj);
```

Listagem 3.2. Trecho da PCA

As saídas da função serão as variáveis:

`pc` – guarda os coeficientes dos componentes principais

`score` – guarda as coordenadas dos dados originais do projeto no novo sistema de coordenadas. Essas coordenadas são definidas pelos componentes principais.

`latente` – vetor contendo a variância. Essa variância é explicada pelo correspondente componente principal.

`tsquared` – de Hotelling T2, é um gráfico de controle multivariado da distância, sendo ela de cada observação partindo-se do centro do conjunto de dados. É eficiente para buscas por pontos mais extremos contidos nos dados.

A linha de código seguinte, dividindo a soma acumulativa pela soma da variável latente, mostrará os componentes mais contribuintes para a variância.

```
cumsum(latente)./sum(latente)
```

```
ans =

    0.3603
    0.5648
    0.6678
    0.7578
    0.8390
    0.9082
    0.9617
    0.9896
    0.9982
    0.9994
    1.0000
```

A saída da função mostrou que dos onze componentes analisados, quatro deles são responsáveis por 75% da variância.

3.4.4 Passo a passo da função princomp

Após a padronização dos dados, inicia-se o processo efetivo de análise de componentes principais checando a covariância entre as variáveis, utilizando a função *cov* do MATLAB®. Essa função calcula a covariância entre as variáveis atribuindo a saída a uma nova variável.

```
CV = cov(dadosaj);
```

Também poderia ser utilizada a seguinte função para a mesma finalidade:

```
CV = corr(dadosaj,dadosaj);
```

Esta saída conterá uma matriz com o grau de interdependência entre cada variável, podendo algumas variáveis serem inter-relacionadas em um nível elevado, maior que 0,70 ou independentes, com covariância igual a zero.

	Area	Momentos	Perímetro	Compacidade	Diametro	Media_Pico	Media_Valas	X	Y	DistanciaM	Simetria
Area	1	0,08924608	0,69771373	-0,737783915	0,479294496	0,077468096	0,712678614	0,03407158	0,31971297	-0,04506642	0,7042714
Momentos	0,08924608	1	0,03718391	-0,643676627	0,118432989	0,091449977	-0,042398687	-0,23872347	-0,13848226	0,02869891	0,5752326
Perímetro	0,69771373	0,03718391	1	-0,293643209	0,950324342	-0,01894509	0,654932738	0,14173887	0,36755637	-0,13889732	0,2996327
Compacidade	-0,7377839	-0,6436766	-0,2936432	1	-0,14862362	-0,14724342	-0,366865351	0,14893202	-0,06787806	-0,03578347	-0,9304051
Diametro	0,4792945	0,11843299	0,95032434	-0,14862362	1	-0,03471681	0,501741623	0,11730971	0,27686139	-0,15254221	0,1588491
Media_pico	0,0774681	0,09144998	-0,0189451	-0,147243424	-0,03471681	1	0,00129722	-0,03265583	0,06657388	0,005432302	0,124634
Media_Valas	0,71267861	-0,0423987	0,65493274	-0,366865351	0,501741623	0,00129722	1	0,15025922	0,47397059	-0,06315286	0,3778379
X	0,03407158	-0,2387235	0,14173887	0,148932019	0,117309714	-0,03265583	0,150259217	1	0,33546778	-0,07371091	-0,1080484
Y	0,31971297	-0,1384823	0,36755637	-0,067878063	0,276861391	0,066573885	0,473970592	0,33546778	1	-0,02290223	0,0744322
DistânciaM	-0,0450664	0,02869891	-0,1388973	-0,035783472	-0,15254221	0,005432302	-0,063152865	-0,07371091	-0,02290223	1	0,0243567
Simetria	0,7042714	0,57523261	0,29963266	-0,930405128	0,158849096	0,124634018	0,377837904	-0,10804837	0,07443218	0,024356731	1

Figura 3.6. Matriz de covariância dos dados.

Obtendo a matriz de covariância, precisaremos fazer a análise discriminante linear, ou seja, comparar as instâncias e verificar a que grupo pertencem. Para isto, calcularemos os autovalores e autovetores, pela função *eig(x)*, sendo x a matriz de covariância adquirida anteriormente. O *autovet* são os vetores associados a cada autovalor.

```
[autoVet,autoVal] = eig(CV);
```

A função acima gera matrizes de autovalores (D) e autovetores (V) da matriz de covariância, de modo que *A*V = V*D*. No entanto na saída da função, os autovalores como podemos observar, ficam disposto de forma ascendente, sendo a última coluna correspondente ao primeiro componente principal. Para corrigirmos isso utilizamos a seguinte função para colocar os autovalores em ordem decrescente, fazendo com que a primeira coluna seja a do primeiro componente principal. OrdAutoVal corresponde à variável *latente* da função *princomp*.

```
[ordAutoVal,indAutoVal] = sort(diag(autoVal),1, ' descend')
```

O *indAutoVal guardará os índices dos componentes.*

Utilizando novamente a função *cumsum*, mas com os autovalores já ordenados. A saída será em ordem crescente dos componentes que mais contribuiram para a variância.

```
cumsum(ordAutoVal)./sum(ordAutoVal)
ans =

    0.3603
    0.5648
    0.6678
    0.7578
    0.8390
    0.9082
    0.9617
    0.9896
    0.9982
    0.9994
    1.0000
```

Pelo resultado obtido acima foi possível concluir que apenas com quatro componentes 75,78% dos dados são explicados.

```
ordAutoVet = autoVet(:,indAutoVal)
```

A nova variável, *ordAutoVet*guardará os coeficientes dos componentes principais, mesma finalidade da variável *pc* da função princomp, passo a passo explicada acima.

E finalizando o processo de análise de componentes principais, *newdados* é a matriz que receberá os dados transformados. Esses novos dados não são modificados, pois eles correspondem aos mesmo dados originais.

```
newdados = dadosaj * ordAutoVet(:,1:4)
```

Capítulo 4
RESULTADOS

Nesta etapa do projeto os dados obtidos até aqui serão analisados, treinados e então classificados. Para a realização deste processo teremos o auxílio do programa Weka juntamente com o MATLAB®.

4.1 WEKA

WEKA é um software criado pela Universidade de Waikato (Nova Zelândia), escrito em linguagem Java™ que usa a GNU General PublicLicense (GPL). Este programa contém uma GUI que permite interagir com arquivos de dados e dá a possibilidade de a partir desses dados produzir resultados visuais para melhor compreensão.

Ao inicializar o software WEKA, o *Weka GUI Chooser* (Selecionador de GUI) é apresentado. Nele estão disponíveis quatro modos de trabalho: *Explorer*, *Experimenter*, *KnowledgeFlow* e*Simple CLI*.

Para este projeto em questão, utilizaremos somente a opção Explorer.

Figura 4.1. Interface do Weka.

Utilizando a opção *Explorer* é apresentado vários painéis, dos quais dão acesso aos principais recursos do programa.

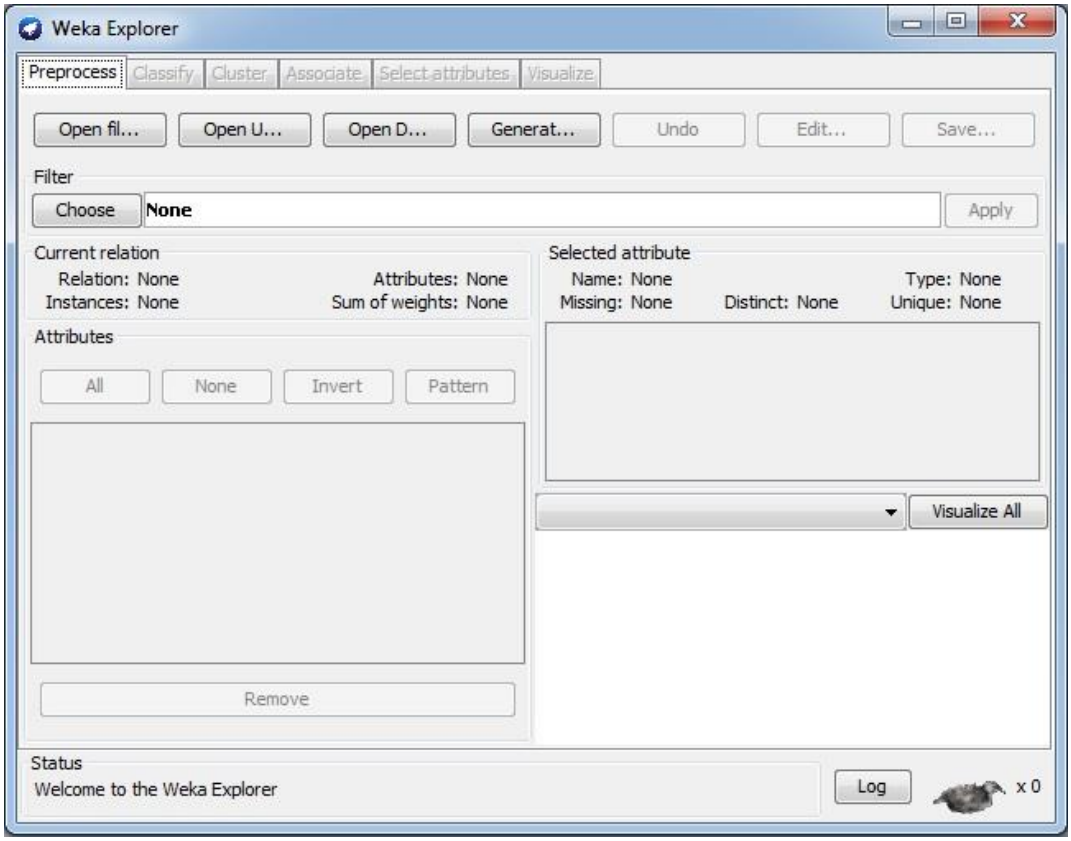


Figura 4.2. Interface do Weka Explorer

4.1.2 Construindo arquivo para o WEKA

Para utilizarmos o WEKA carregando dados, é necessário criar um arquivo que esteja em um formato entendido para o programa. O principal método usado é o formato de Arquivo de Atributo-Relação (ARFF). Neste arquivo declaramos atributos e seus tipos específicos. Podemos definir em cada coluna o que ela contém, sendo os dados de cada coluna separados por vírgulas.

```
@RELATION folha

@ATTRIBUTE area            NUMERIC @ATTRIBUTE momentos NUMERIC
@ATTRIBUTE perimetro       NUMERIC @ATTRIBUTE compacidade
      NUMERIC
@ATTRIBUTE diametro NUMERIC @ATTRIBUTE pediapico       NUMERIC
@ATTRIBUTE mediavalas      NUMERIC @ATTRIBUTE coordx   NUMERIC
@ATTRIBUTE coordy   NUMERIC @ATTRIBUTE distmedia       NUMERIC
@ATTRIBUTE simetria NUMERIC
@ATTRIBUTE class    {Folha-pitanga,Folha-coquecasa,Folha-jequetiba,
Folha-lima,Folha-limao,Folha-desc1,Folha-desc2,Folha-desc3,Folha-
desc4,Folha-desc5}
@DATA
9179,0.0197,756,62.2656,375.012,0.0011,-
0.0049,246.5676,257.996,96.6972,0.5626
```

```
9317,0.0197,756,61.3434,375.192,8.1494,-
0.0043,255.828,247.996,96.7768,0.5124
10527,0.0232,851,68.7946,423.1182,-1.5136,-
0.004,255.8884,245.7415,108.6469,0.491
```

Listagem 4.1. Trecho do arquivo folhas.arff

4.1.3 Carregando Dados no WEKA

A aba Preprocess nos auxilia no pré-processamento de dados, importando as informações a partir de um banco de dados. Este banco pode ser carregado por um arquivo com extensão .CSV, .ARFF, etc.

Neste projeto foi utilizado o arquivo *dados_com_pca.arff*, contendo os dados de todas as folhas analisadas aplicadas na PCA. Após carregar o arquivo, o programa nos permite que revisemos os dados, na parte à esquerda da janela do WEKA Explorer pode-se observar todas as colunas de dados e o número de linhas, sendo elas respectivamente atributos e instâncias.

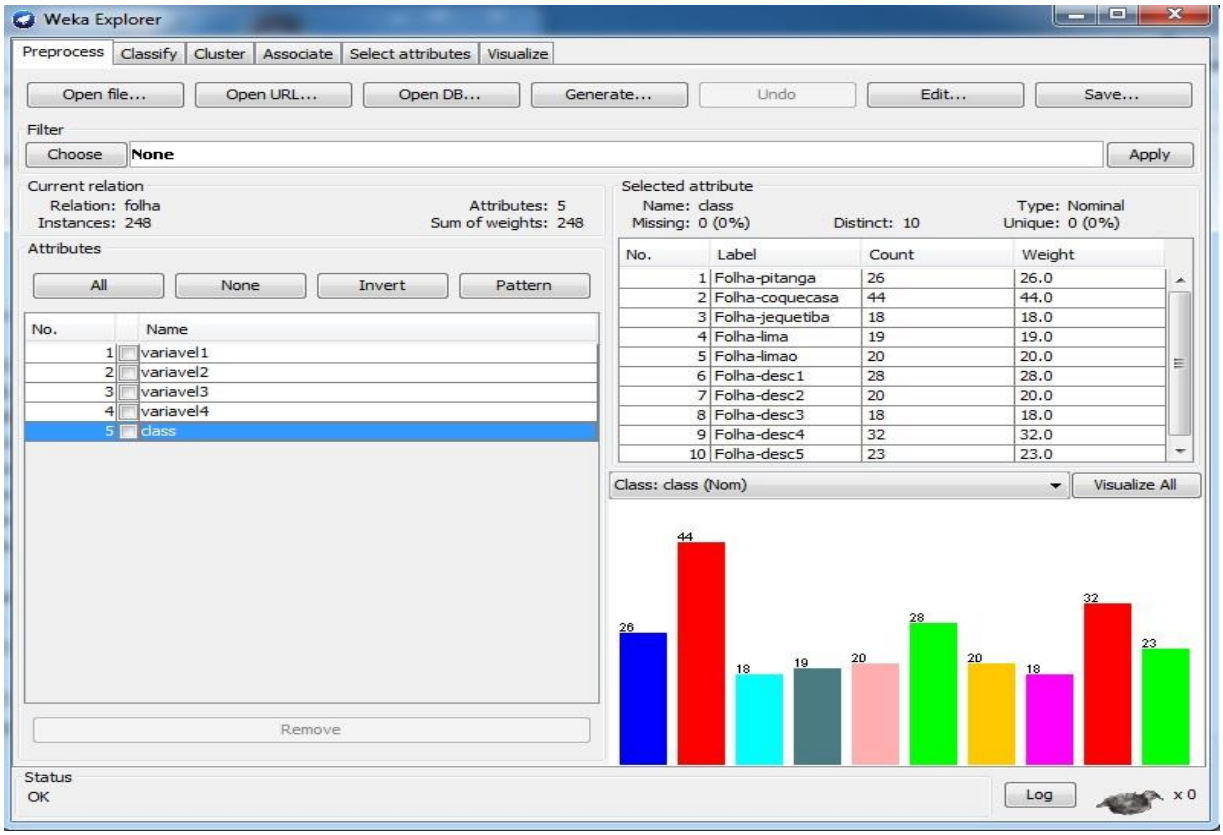


Figura 4.3. Visualização dos dados.

A cada coluna selecionada, informações sobre ela são expostas. À direita tem-se informações adicionais de cada atributo, como o maior e menor valor, média e desvio padrão.

Para visualizar os dados e examiná-los basta clicar no botão *Visualize All*.

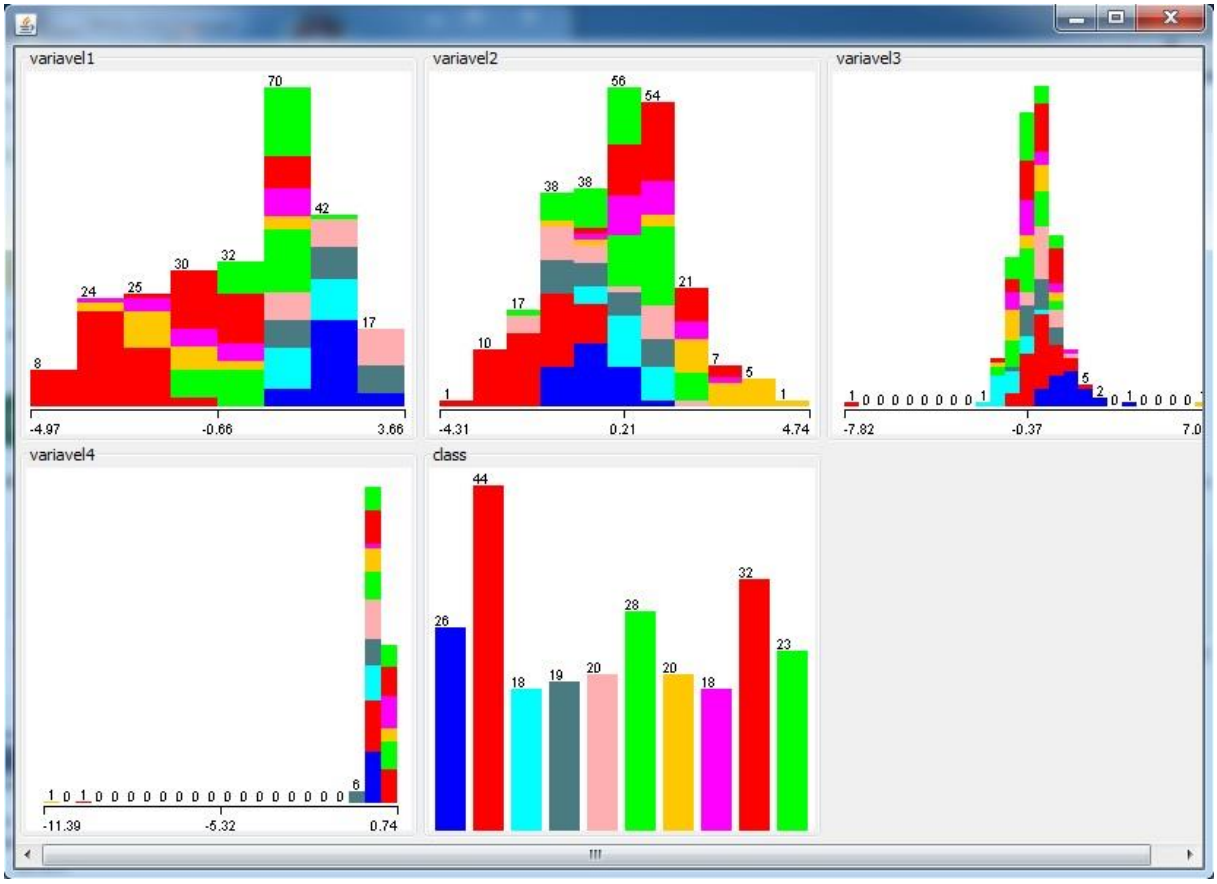


Figura 4.4. Visualização dos atributos e instâncias no WEKA.

4.2 CRIANDO MODELO

Para criamos um modelo primeiramente abriremos a aba *Classify*, logo após o botão *choose* e selecionar o modelo desejado.

Após escolhermos o modelo, apontamos para o programa quais dados deve usar, pois, há diferentes opções para usar, tais como:

- Use training set – usa os dados do arquivo *.arff* fornecido.
- Suppliedtest set – usa um diferente conjunto dados para a construção do modelo;
- Cross-validation – validação cruzada, baseado em subconjuntos de dados, para criar o modelo final calcula a média;
- Percentage Split – utiliza a porcentual obtido por um subconjunto de dados.

Os algoritmos escolhidos para comparação para a classificação dos dados recolhidos foram o *J48* e o *LMT*, usando a configuração original de ambos.

Para o conjunto de teste optamos por utilizar os valores transformados pela PCA, pelo fato de possuir menos variáveis, tornando o processo de classificação menos complexo.

Utilizamos para a análise os dados fornecidos pelo arquivo *dados_com_pca.arff,* a opção de teste use training test.

4.3 RESULTADOS PRINCIPAIS

J48

```
Correctly Classified Instances         222               89.5161%
Incorrectly Classified Instances        26               10.4839%
```

Classe	Precisão	Recall	F-measure
Pitanga	0,926	0,962	0,943
Coquecasa	1,000	0,977	0,989
Jequitiba	0,895	0,944	0,919
Lima	1,000	0,842	0,914
Limão	0,818	0,900	0,857
Desconhecido 1	0,800	0,857	0,828
Desconhecido 2	0,895	0,850	0,872
Desconhecido 3	0,846	0,611	0,710
Desconhecido 4	0,875	0,875	0,875
Desconhecido 5	0,852	1,000	0,920

Tabela 4.1: Acurácia do algoritmo J48 em cada classe.

```
=== Confusion Matrix ===

  a  b  c  d  e  f  g  h  i  j   <-- classified as
 25  0  0  0  0  0  0  0  0  1 |  a = Folha-pitanga
  0 43  0  0  0  0  0  1  0  0 |  b = Folha-coquecasa
  0  0 17  0  1  0  0  0  0  0 |  c = Folha-jequetiba
  1  0  0 16  2  0  0  0  0  0 |  d = Folha-lima
  1  0  0  0 18  1  0  0  0  0 |  e = Folha-limao
  0  0  0  0  0 24  0  0  4  0 |  f = Folha-desc1
  0  0  0  0  1  1 17  1  0  0 |  g = Folha-desc2
  0  0  0  0  0  4  2 11  0  1 |  h = Folha-desc3
  0  0  2  0  0  0  0  0 28  2 |  i = Folha-desc4
  0  0  0  0  0  0  0  0  0 23 |  j = Folha-desc5
```

Figura 4.5. Matriz de confusão utilizando o algoritmo J48.

LMT

```
Correctly Classified Instances         224               90.3226%
Incorrectly Classified Instances        24                9.6774%
```

Classe	Precisão	Recall	F-measure

Pitanga	0,833	0,962	0,926
Coquecasa	1,000	1,000	1,000
Jequitiba	1,000	1,000	1,000
Lima	0,850	0,895	0,872
Limão	0,882	0,750	0,811
Desconhecido 1	0,867	0,929	0,897
Desconhecido 2	0,947	0,900	0,923
Desconhecido 3	0,929	0,722	0,813
Desconhecido 4	0,765	0,813	0,788
Desconhecido 5	0,917	0,957	0,936

Tabela 4.2: Acurácia do algoritmo LMT em cada classe.

```
=== Confusion Matrix ===

  a  b  c  d  e  f  g  h  i  j   <-- classified as
 25  0  0  1  0  0  0  0  0  0 |  a = Folha-pitanga
  0 44  0  0  0  0  0  0  0  0 |  b = Folha-coquecasa
  0  0 18  0  0  0  0  0  0  0 |  c = Folha-jequetiba
  0  0  0 17  2  0  0  0  0  0 |  d = Folha-lima
  3  0  0  2 15  0  0  0  0  0 |  e = Folha-limao
  0  0  0  0  0 26  0  0  2  0 |  f = Folha-desc1
  0  0  0  0  0  1 18  0  1  0 |  g = Folha-desc2
  0  0  0  0  0  0  0 13  4  1 |  h = Folha-desc3
  0  0  0  0  0  3  1  1 26  1 |  i = Folha-desc4
  0  0  0  0  0  0  0  0  1 22 |  j = Folha-desc5
```

Figura 4.6. Matriz de confusão utilizando o algoritmo LMT.

NaiveBayes

```
Correctly Classified Instances         175    70.5645%
Incorrectly Classified Instances        73    29.4355%
```

Classe	Precisão	Recall	F-measure
Pitanga	0,828	0,923	0,873
Coquecasa	1,000	0,977	0,989
Jequitiba	0,667	0,889	0,762
Lima	0,889	0,421	0,571
Limão	0,632	0,600	0,615
Desconhecido 1	0,517	0,536	0,526
Desconhecido 2	0,833	0,500	0,625
Desconhecido 3	0,667	0,444	0,533
Desconhecido 4	0,528	0,594	0,559
Desconhecido 5	0,727	0,706	0,700

Tabela 4.3: Acurácia do algoritmo NaiveBayes em cada classe.

```
=== Confusion Matrix ===

  a  b  c  d  e  f  g  h  i  j   <-- classified as
 24  0  0  0  2  0  0  0  0  0 |  a = Folha-pitanga
  0 43  0  0  0  0  1  0  0  0 |  b = Folha-coquecasa
  0  0 16  0  0  0  0  0  0  2 |  c = Folha-jequetiba
  3  0  3  8  5  0  0  0  0  0 |  d = Folha-lima
  2  0  2  1 12  0  0  0  3  0 |  e = Folha-limao
  0  0  1  0  0 15  0  3  6  3 |  f = Folha-desc1
  0  0  0  0  0  3 10  0  5  2 |  g = Folha-desc2
  0  0  0  0  0  3  1  8  2  4 |  h = Folha-desc3
  0  0  2  0  0  6  0  1 19  4 |  i = Folha-desc4
  0  0  0  0  0  2  0  0  1 20 |  j = Folha-desc5
```

Figura 4.7. Matriz de confusão utilizando o algoritmo NaiveBayes.

IBK

```
Correctly Classified Instances         248      100%
Incorrectly Classified Instances         0        0%
```

Classe	Precisão	Recall	F-measure
Pitanga	1,000	1,000	1,000
Coquecasa	1,000	1,000	1,000
Jequitiba	1,000	1,000	1,000
Lima	1,000	1,000	1,000
Limão	1,000	1,000	1,000
Desconhecido 1	1,000	1,000	1,000
Desconhecido 2	1,000	1,000	1,000
Desconhecido 3	1,000	1,000	1,000
Desconhecido 4	1,000	1,000	1,000
Desconhecido 5	1,000	1,000	1,000

Tabela 4.4: Acurácia do algoritmo IBK em cada classe.

```
=== Confusion Matrix ===

  a  b  c  d  e  f  g  h  i  j   <-- classified as
 26  0  0  0  0  0  0  0  0  0 |  a = Folha-pitanga
  0 44  0  0  0  0  0  0  0  0 |  b = Folha-coquecasa
  0  0 18  0  0  0  0  0  0  0 |  c = Folha-jequetiba
  0  0  0 19  0  0  0  0  0  0 |  d = Folha-lima
  0  0  0  0 20  0  0  0  0  0 |  e = Folha-limao
  0  0  0  0  0 28  0  0  0  0 |  f = Folha-desc1
  0  0  0  0  0  0 20  0  0  0 |  g = Folha-desc2
  0  0  0  0  0  0  0 18  0  0 |  h = Folha-desc3
  0  0  0  0  0  0  0  0 32  0 |  i = Folha-desc4
  0  0  0  0  0  0  0  0  0 23 |  j = Folha-desc5
```

Figura 4.8. Matriz de confusão utilizando o algoritmo IBK.

4.4 ANALISANDO RESULTADOS

A partir dos resultados obtidos podemos observar que, em função da porcentagem de acerto, a do algoritmo *IBK* foi de 100%, seguido pelo algoritmo *LMT*, com 90,32%, J48 com 89,52% e o *NaiveBayes* com 70,56%. Optamos por não constar as opções de *Cross-validation* e a percentagem Split devido às porcentagens de acertos serem insatisfatórias. As duas configurações: 50% para treino e 50% teste e 75% treino e 25% para teste não alcançaram 70%.

Em relação ao tempo de aprendizagem dos algoritmos *IBK*, *LMT, J48* e *NaiveBayes* em segundos, foram respectivamente de 0.03, 0, 0 e 0.09.

Comparando o desempenho entre os algoritmos de árvore de decisão, verificamos que o *LMT*, em relação à porcentagem de acerto, obteve um resultado melhor que o *J48*, com duas instâncias à mais classificadas corretamente, como pudemos analisar pela matriz de confusão mostrada anteriormente.

Considerando a construção da árvore de decisão, o algoritmo LMT mostrou-se mais compacta, subdividindo os grupos de dados em um número muito menor de nós necessários.

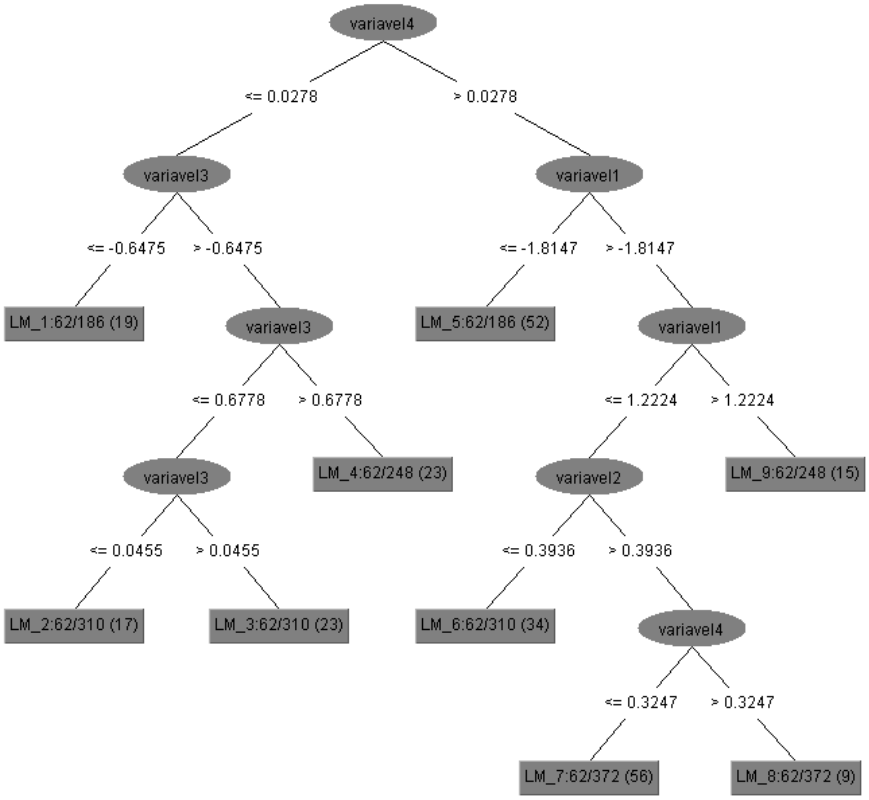


Figura 4.9. Visualização da Árvore de decisão construída pelo algoritmo LMT.

Após todos os dados computados, analisados e classificados, pudemos observar e concluir que o algoritmo asseado em instâncias, o IBK, foi o único que conseguiu classificar todas os dados com porcentagem de acerto de 100%, agrupando corretamente todas as 248 instâncias utilizadas para estudo.

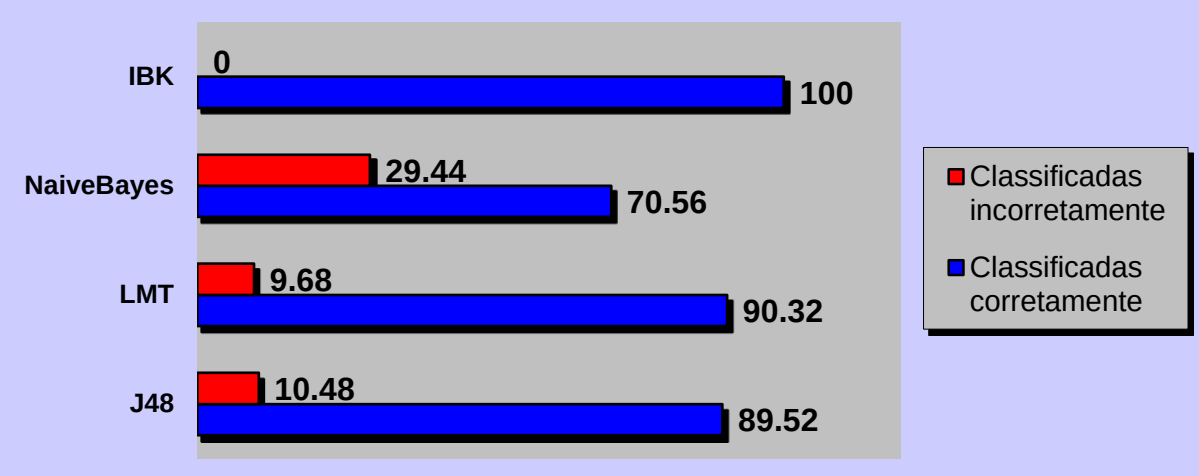


Figura 4.10. Percentual de acertos de casa modelo.

A precisão, recall e F-Measure, que é calculada por 2*Precisão*Recall/(Precisão+Recall), também foram superiores.

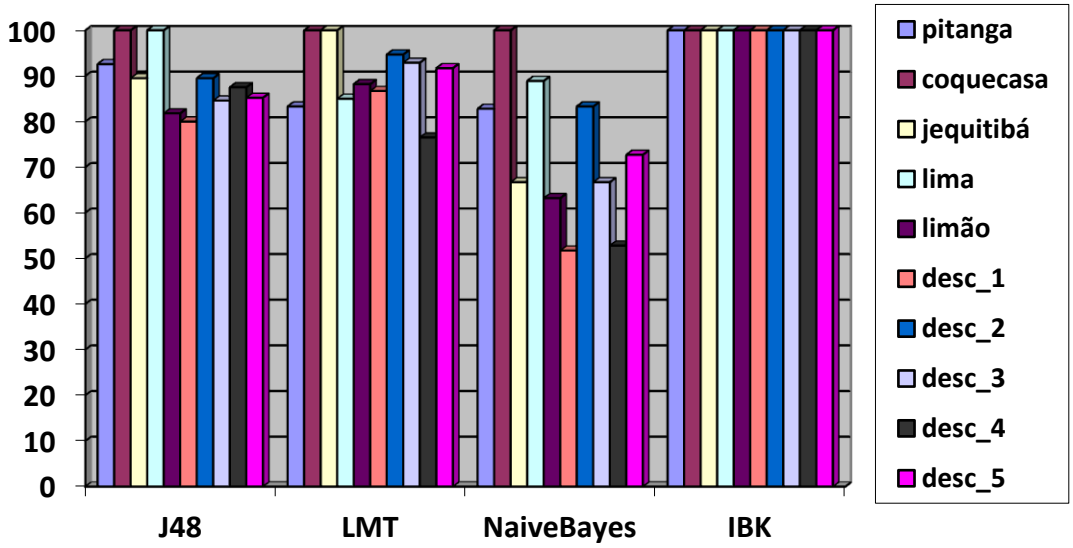


Figura 4.11. Gráfico de precisão dos modelos de aprendizagem.

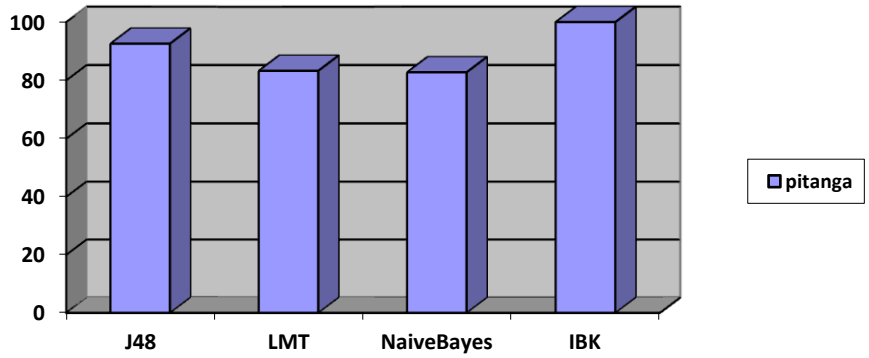


Figura 4.12. Gráfico de precisão dos quatros modelos para a folha de pitanga.

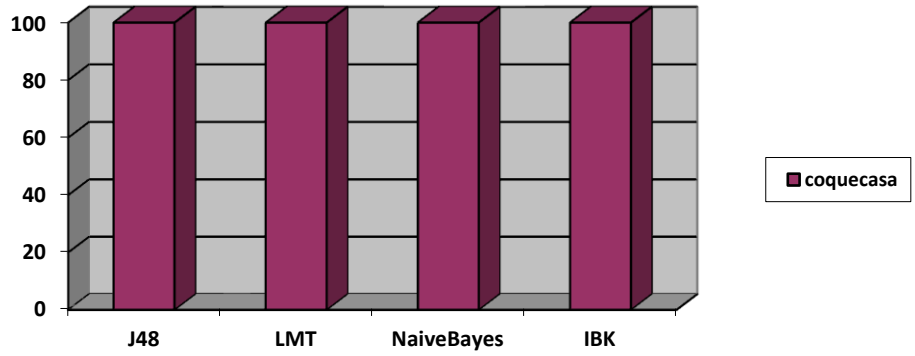


Figura 4.13. Gráfico de precisão dos quatros modelos para a folha de coquecasa.

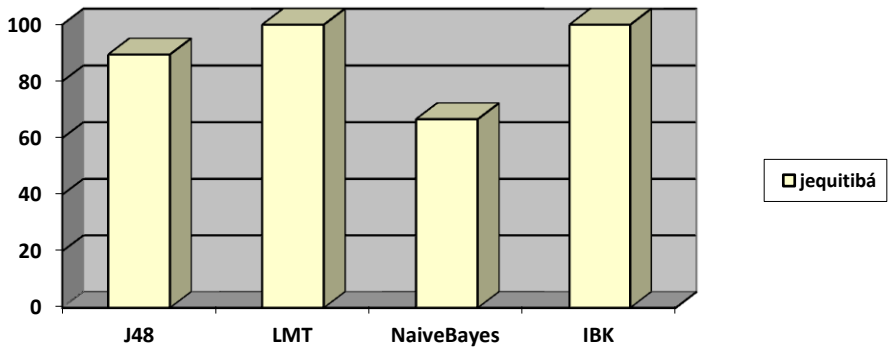


Figura 4.14. Gráfico de precisão dos quatros modelos para a folha de jequitibá.

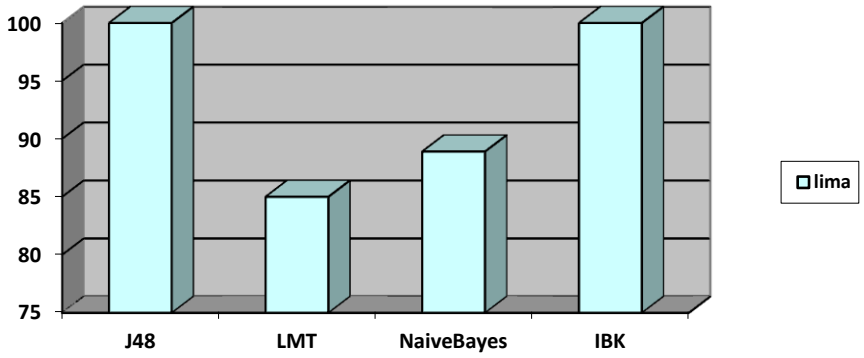


Figura 4.15. Gráfico de precisão dos quatros modelos para a folha de lima.

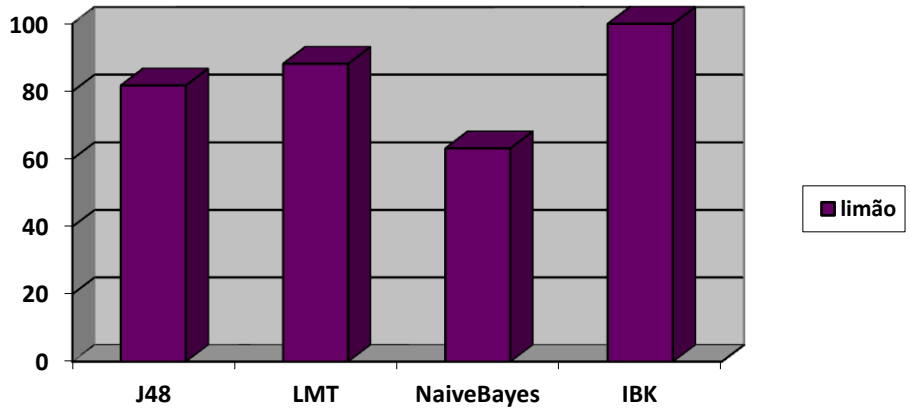


Figura 4.16. Gráfico de precisão dos quatros modelos para a folha de limão.

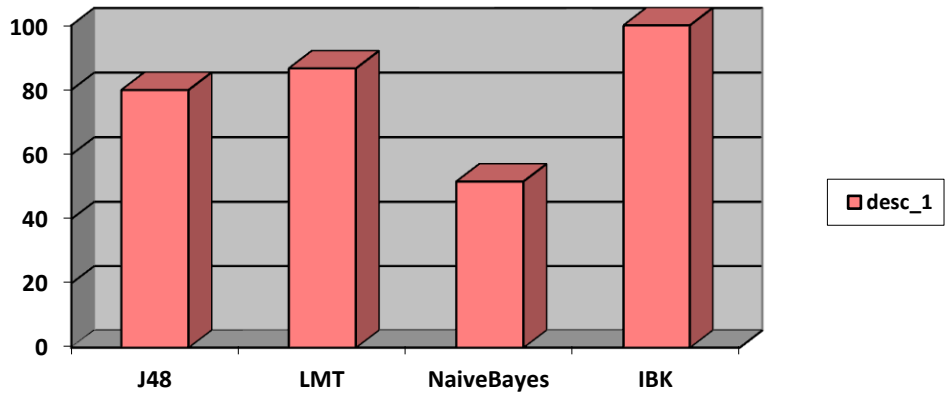


Figura 4.17. Gráfico de precisão dos quatros modelos para a folha desconhecida_1.

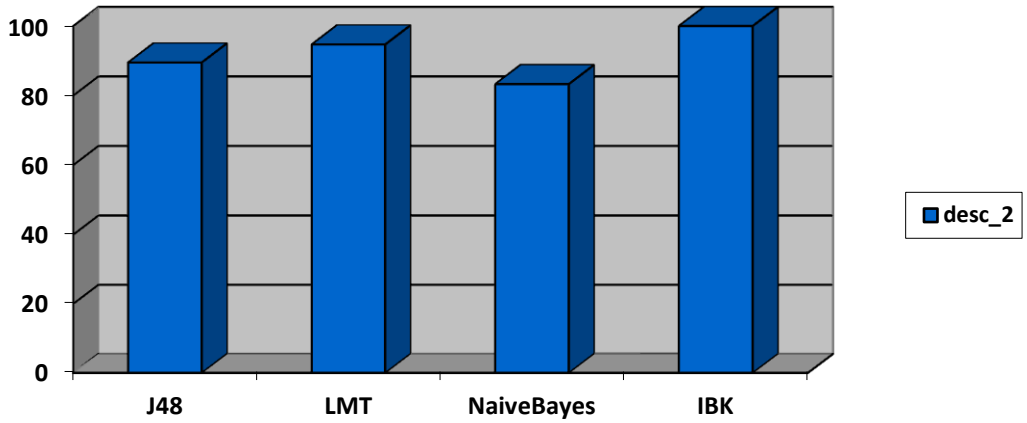


Figura 4.18. Gráfico de precisão dos quatros modelos para a folha desconhecida_2.

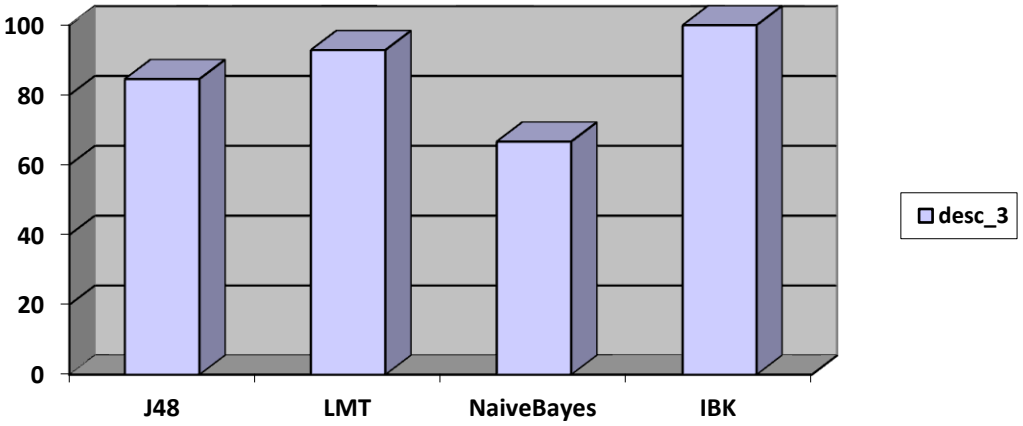


Figura 4.19. Gráfico de precisão dos quatros modelos para a folha desconhecida_3.

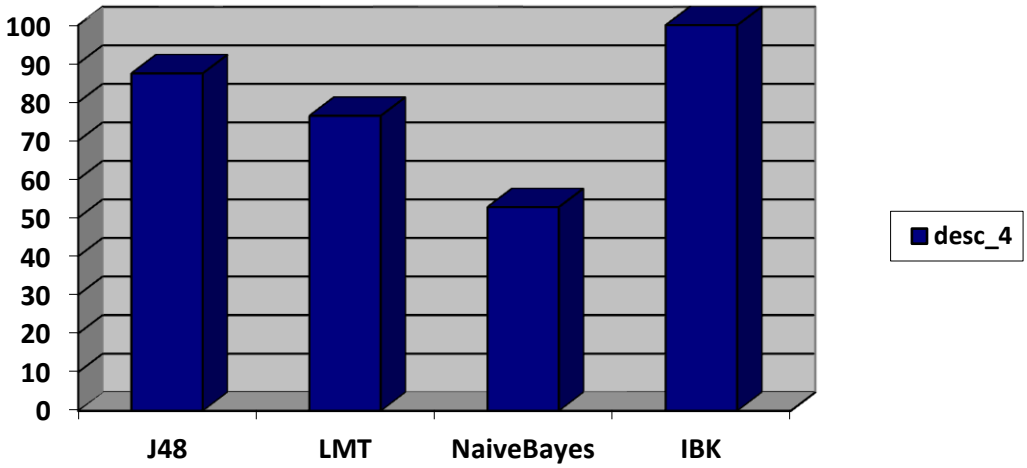


Figura 4.20. Gráfico de precisão dos quatros modelos para a folha desconhecida_4.

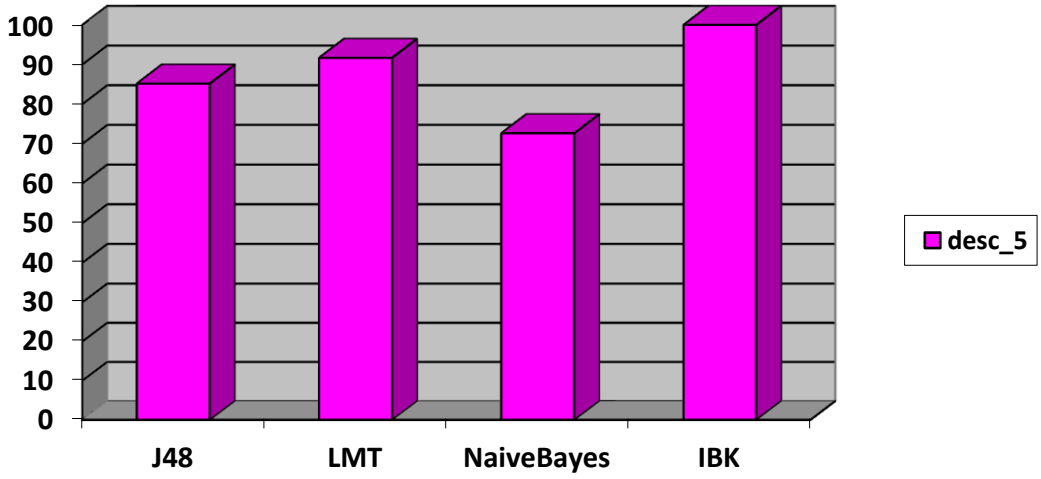


Figura 4.21. Gráfico de precisão dos quatros modelos para a folha desconhecida_5.

Figura 4.22. Gráfico de precisão dos quatros modelos e suas porcentagens.

A estatística *Kappa*, que calcula o acordo de previsão da instância com a verdadeira classe, também foi superior a todos os outros algoritmos, com valor igual a 1, ou seja, em total acordo. A matriz de confusão (ou tabela de continência) do IBK com as dez classes, e, portanto, uma matriz 10x10, o número de instâncias corretamente classificadas resultou em 248 casos. Portanto, 100% de acerto.

Capítulo 5
CONCLUSÃO

Na elaboração deste projeto passamos por diversos passos até realizarmos a classificação. O pré-processamento de imagens consistiu em boa parte das transformações lineares e não lineares aplicadas à imagem. Envolveu processos de segmentação, binarização e extração de características de imagens para segmentação da mesma. As informações que nos interessavam para posteriormente utilizá-las em outros meios, como por exemplo, o cálculo do perímetro, área, simetria, dentre outros, puderam ser encontradas e representadas de uma forma diferente, por gráficos em barra, tabelas de dispersão de variáveis, etc.

Com todos os dados extraídos, seguimos com o trabalho verificando todas as variáveis e instâncias existente, para então analisarmos seus componentes principais (PCA). Este procedimento utilizou transformações com auxílio de várias operações matemáticas, como a apuração de seus autovalores e autovetores, com o objetivo de diminuir a complexidade de um conjunto de dados com muitos atributos, a fim de encontrar seus componentes principais e então facilitar a próxima etapa do projeto. Apesar do número de componentes ser menor que no conjunto de dados originais, a essência dos dados continua a mesa.

Para a classificação dos dados do projeto testamos quatro algoritmos de aprendizado de máquina para classificar duzentas e quarenta e oito folhas da pasta de arquivos de folhas da Mata Atlântica. A partir de um arquivo .arff contendo os dados já transformados pela PCA, comparamos os modelos de aprendizado, realizando a análise da porcentagem de acerto, erro e precisão, verificando o mais eficiente para o conjunto de dados de interesse, que no caso deste estudo em questão, o algoritmo baseado em instancias, o IBK, se sobressaiu sobre os demais, classificando corretamente todas as instâncias contidas no conjunto de teste.

No estudo que fizemos concluímos que, a classificação de folhas usando medidas invariantes é um processo que requer a execução de muitos passos. Dentre eles, um bom pré-processamento da imagem é fundamental para podermos extrair todos os dados necessários corretamente e seguirmos sem dificuldades em aplicarmos outros processos, como a mineração de dados, que além de diminuir a complexidade de análise dos elementos, dá a possibilidade de examinarmos de forma a conseguirmos enxergar o resultado sobre alguns atributos mais evidentemente, Sejam eles através de gráficos ou tabelas.

Portanto, concluímos que para obtermos uma classificação com bons índices, precisamos decidir qual a melhor forma de testá-los, pois nem sempre os modelos escolhidos

para uma proposta funcionam para outras. Testá-las é essencial e a sua comparação também é importante para uma porcentagem maior de acerto.

5.1 PROJETOS FUTUROS

Existem muitas maneiras de realizar classificação de um grupo de dados. Em especial, para uma classificação mais detalhada, poderiam ser utilizadas outras técnicas, aplicando a convolução com uma gaussiana na imagem e variando as escalas, criando então imagens mais suavizadas, para posteriormente extrair dados das mesmas, aplicando assim os conceitos da estatística multivariada, mineração de dados e aprendizagem de máquina.

6 REFERÊNCIAS BIBLIOGRÁFICAS

BOLDRINI, José Luiz e COSTA, Sueli Rodrigues e FIGUEIREDO, Vera Lúcia e WETZLER, G. Henry – *Álgebra Linear, 3ª edição*, Harper &Row do Brasil, São Paulo, 1980.

CASANOVA, Dalcimar. *Identificação de espécies vegetais por meio da analise de textura foliar*.

CESAR, Roberto Marcondes Jr e COSTA, Luciano da Fontoura. *Shape classification and analysis: theory and practice*, 2ª edição. Taylor & Francis Group. 2009.

FAYYAD, Usama; PIATETSKY-SHAPIRO, Gregory; SMYTH, Padhraic. *From data mining to knowledge discovery: An overview*. Cambridge, Massachusetts, and London, England, 1996.

FARIA, Diogo. *Análise e Processamento de Imagem*. 2010. Trabalho Prático (Mestrado em Engenharia Biomédica) – Faculdade de Engenharia, Universidade do Porto.

HELENE, Otaviano. *Método dos Mínimos quadrados com Formalismo Matricial*, 2ª edição Revisada e Ampliada, 2006.

JOLLIFFE, I. T., *Principal Component Analysis*, 2nd edition, Springer, 2002.

KHATTREE, R. & NAIK, D.N. *Multivariate data reduction and discrimination with SAS software*. Cary, NC, USA: SAS Institute Inc., 2000. p.558.

LIAO, P.G; CHEN, T.S & CHUNG, P.C. *A Fast Algorithm for Multilevel Thresholding*, 2001.

LINDSAY, I. S. *A tutorial on Principal Components Analysis*.February 26, 2002.

MARQUES FILHO, Ogê; VIEIRA NETO, Hugo. *Processamento Digital de Imagens*. Rio de Janeiro, Brasport, 1999.

MELLO, C. Alexandre, *Álgebra Linea,Autovalores e Autovetores*.

NICOLIELLO; Heitor, *Uma ferramenta morfométrica MAC 499 - Trabalho de formatura supervisionado.*2007.

RASBAND, S. N. *"Fractal Dimension." Ch. 4 in Chaotic Dynamics of Nonlinear Systems.* New York: Wiley, p. 71-83, 1990.

SOUZA, A. M. *Aula – Correlação linear*. 2001. Disponível em: http://w3.ufsm.br/adriano/aulas/coreg/Aula%2001%20Correla%E7ao%20Linear.pdf. Acessado em 25 de outubro de 2014.

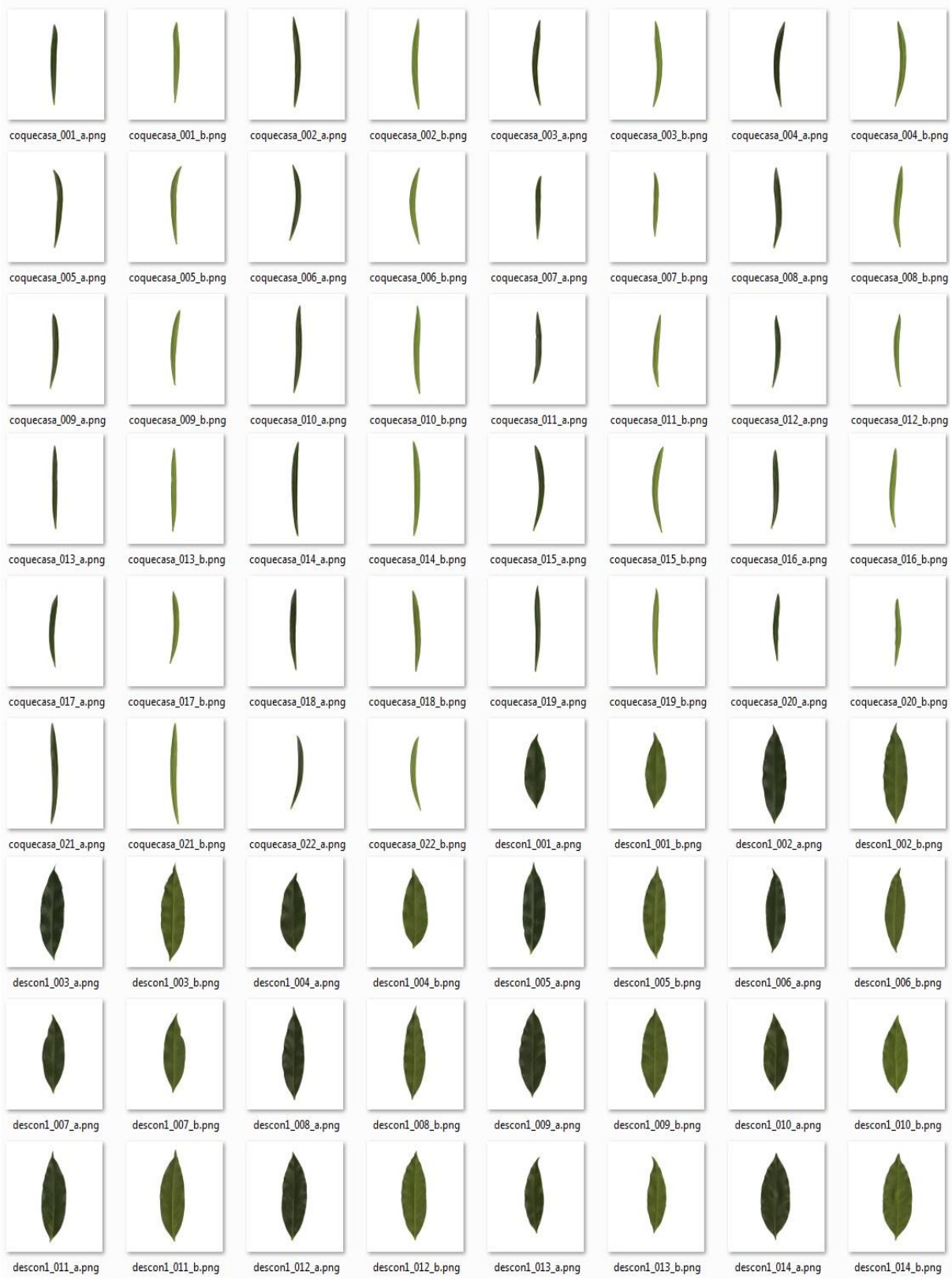
VASCONCELOS, Simone. *Analise de componentes principais*. Disponível em: http://www2.ic.uff.br/~aconci/PCA-ACP.pdf. Acesso em 15 de julho de 2014.

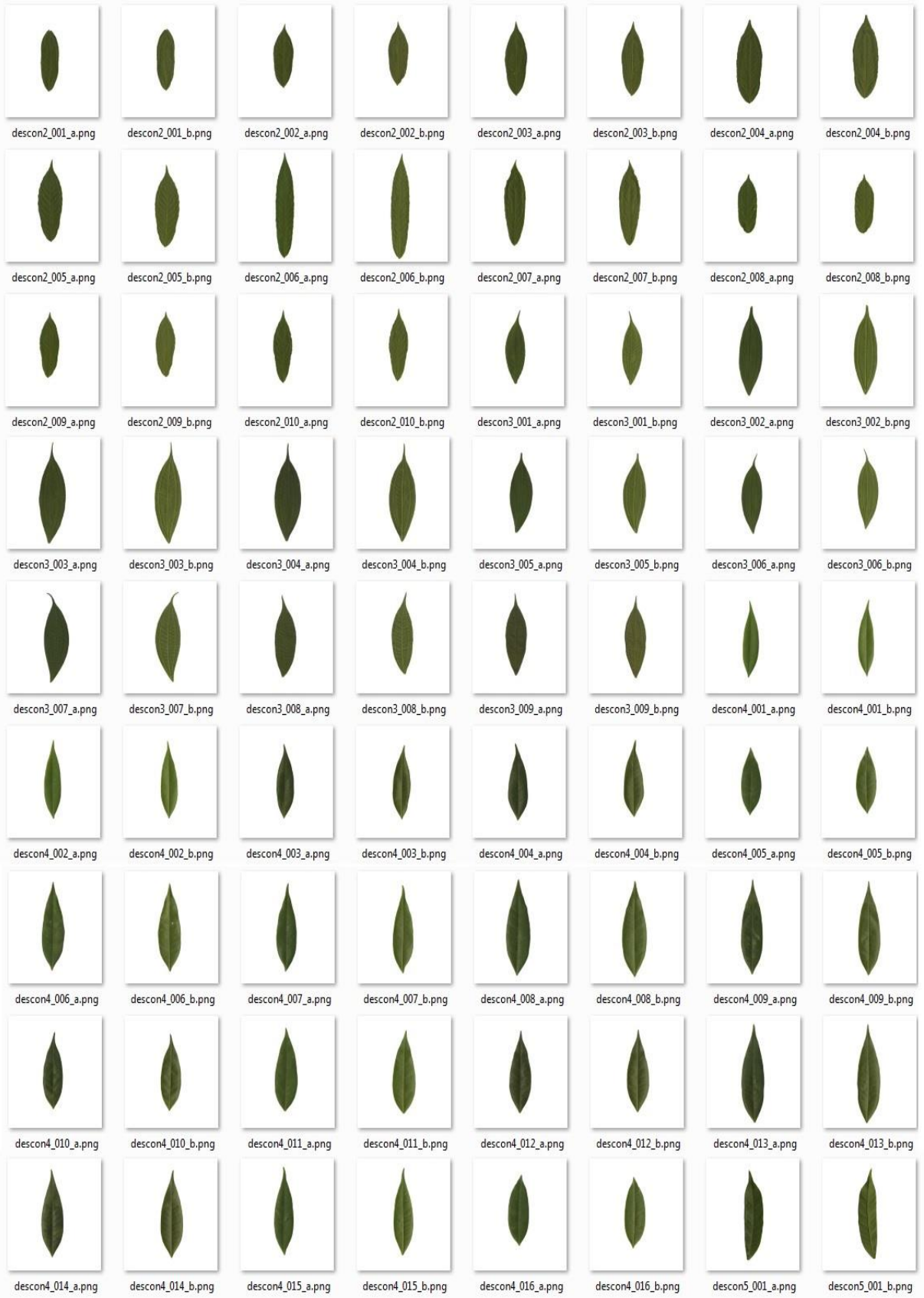
YANG, L. et al. *An improved median-based Otsu image thresholding Algorithm*,2012, China. p.468-469.

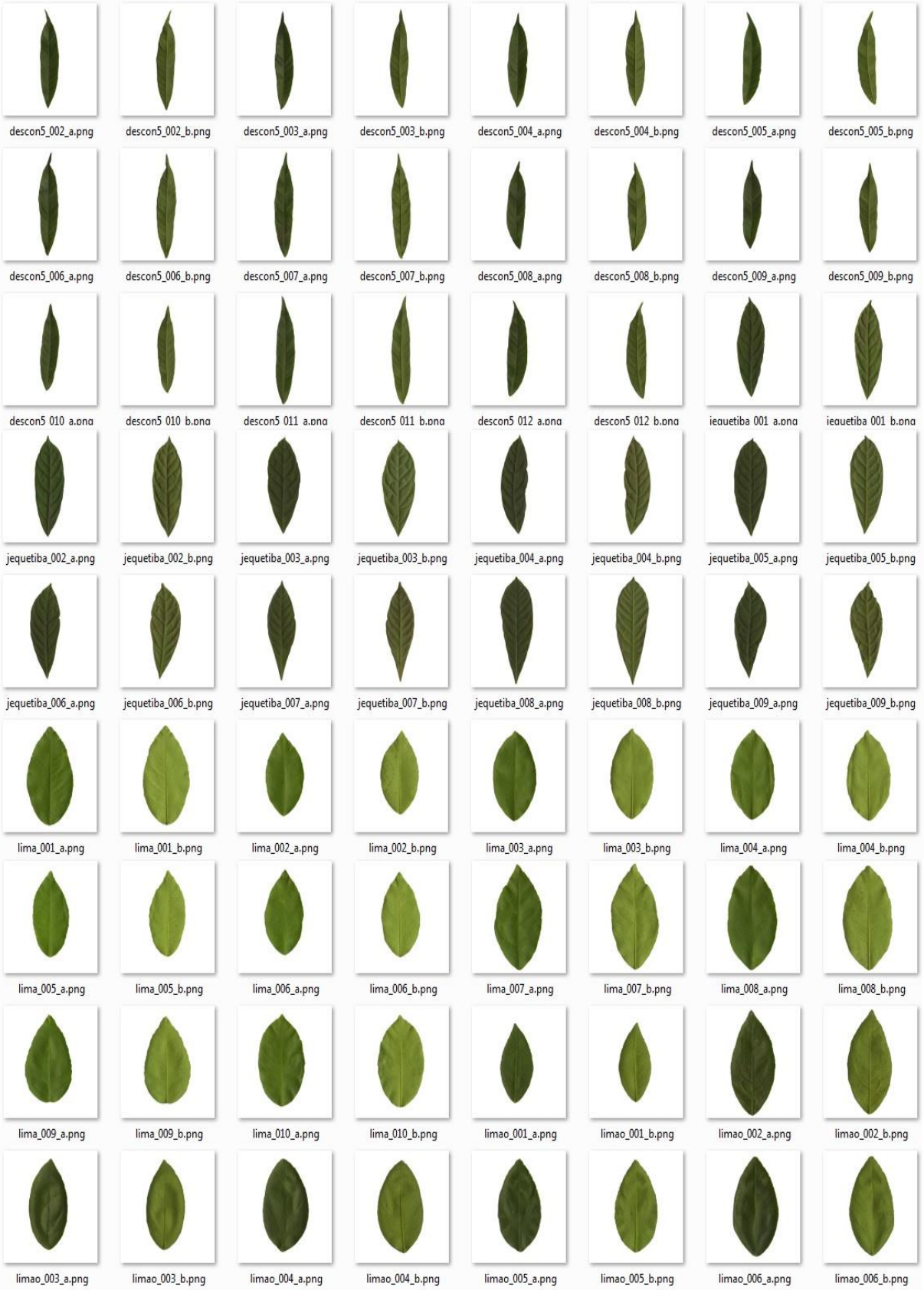
FERREIRA, Jossana. Álgebra Linear,Módulo 2, Autovalores e autovetores Transformações lineares Formas quádricas, Editora da Universidade Federal do Rio Grande do Norte – EDUFRN,2012

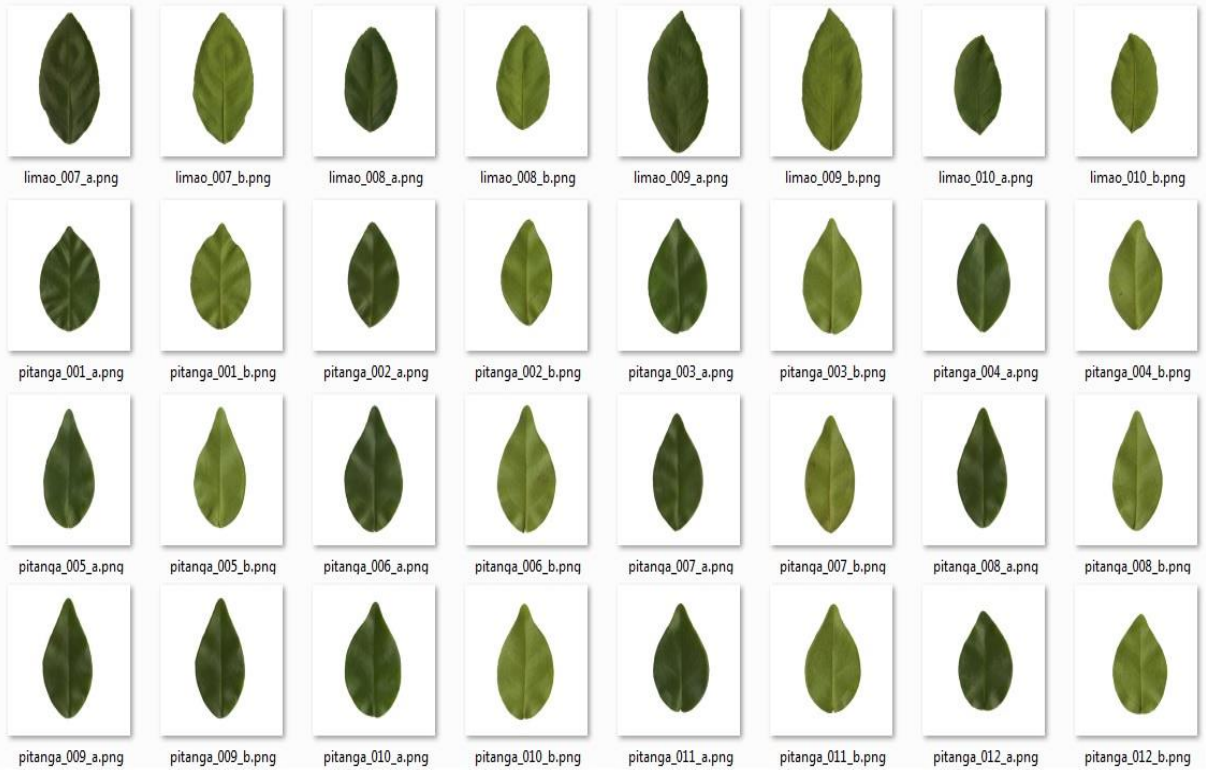
ANEXO A

Folhas do banco de folhas.




descon2_001_a.png
descon2_001_b.png
descon2_002_a.png
descon2_002_b.png
descon2_003_a.png
descon2_003_b.png
descon2_004_a.png
descon2_004_b.png
descon2_005_a.png
descon2_005_b.png
descon2_006_a.png
descon2_006_b.png
descon2_007_a.png
descon2_007_b.png
descon2_008_a.png
descon2_008_b.png
descon2_009_a.png
descon2_009_b.png
descon2_010_a.png
descon2_010_b.png
descon3_001_a.png
descon3_001_b.png
descon3_002_a.png
descon3_002_b.png
descon3_003_a.png
descon3_003_b.png
descon3_004_a.png
descon3_004_b.png
descon3_005_a.png
descon3_005_b.png
descon3_006_a.png
descon3_006_b.png
descon3_007_a.png
descon3_007_b.png
descon3_008_a.png
descon3_008_b.png
descon3_009_a.png
descon3_009_b.png
descon4_001_a.png
descon4_001_b.png
descon4_002_a.png
descon4_002_b.png
descon4_003_a.png
descon4_003_b.png
descon4_004_a.png
descon4_004_b.png
descon4_005_a.png
descon4_005_b.png
descon4_006_a.png
descon4_006_b.png
descon4_007_a.png
descon4_007_b.png
descon4_008_a.png
descon4_008_b.png
descon4_009_a.png
descon4_009_b.png
descon4_010_a.png
descon4_010_b.png
descon4_011_a.png
descon4_011_b.png
descon4_012_a.png
descon4_012_b.png
descon4_013_a.png
descon4_013_b.png
descon4_014_a.png
descon4_014_b.png
descon4_015_a.png
descon4_015_b.png
descon4_016_a.png
descon4_016_b.png
descon5_001_a.png
descon5_001_b.png


descon5_002_a.png
descon5_002_b.png
descon5_003_a.png
descon5_003_b.png
descon5_004_a.png
descon5_004_b.png
descon5_005_a.png
descon5_005_b.png
descon5_006_a.png
descon5_006_b.png
descon5_007_a.png
descon5_007_b.png
descon5_008_a.png
descon5_008_b.png
descon5_009_a.png
descon5_009_b.png
descon5_010_a.png
descon5_010_b.png
descon5_011_a.png
descon5_011_b.png
descon5_012_a.png
descon5_012_b.png
jequetiba_001_a.png
jequetiba_001_b.png
jequetiba_002_a.png
jequetiba_002_b.png
jequetiba_003_a.png
jequetiba_003_b.png
jequetiba_004_a.png
jequetiba_004_b.png
jequetiba_005_a.png
jequetiba_005_b.png
jequetiba_006_a.png
jequetiba_006_b.png
jequetiba_007_a.png
jequetiba_007_b.png
jequetiba_008_a.png
jequetiba_008_b.png
jequetiba_009_a.png
jequetiba_009_b.png
lima_001_a.png
lima_001_b.png
lima_002_a.png
lima_002_b.png
lima_003_a.png
lima_003_b.png
lima_004_a.png
lima_004_b.png
lima_005_a.png
lima_005_b.png
lima_006_a.png
lima_006_b.png
lima_007_a.png
lima_007_b.png
lima_008_a.png
lima_008_b.png
lima_009_a.png
lima_009_b.png
lima_010_a.png
lima_010_b.png
limao_001_a.png
limao_001_b.png
limao_002_a.png
limao_002_b.png
limao_003_a.png
limao_003_b.png
limao_004_a.png
limao_004_b.png
limao_005_a.png
limao_005_b.png
limao_006_a.png
limao_006_b.png



1.A. Algoritmo para encontrar o Contorno
Algoritmo baseado no código de: CESAR, Roberto Marcondes Jr e COSTA, Luciano da Fontoura. *Shapeclassificationandanalysis*. 2001. p.341 – 347.

```
function E = contour(img) %Utilizando método do ceguinho

%dimensoes da matriz
[r,s] = size(img);

%Procura pelo 1o. ponto do contorno
[m,n] = find(img');
m = m(1)-1; n = n(1);

%Representacao da posicao inicial como um sinal complexo
E(1) = m + 1j*n;
n = 2;
dpc = 4;
bol = 0;

%Calculando o 2o. pixel - o algoritmo busca os pelos vizinhos pelo ponto
while(~bol)
de = dpc;
di  =mod(dpc + 1, 8);
Pe = chainpoint( E(1), de); %retorna o vizinho na de(variável) direção
                                %do código da cadeia.
Pi = chainpoint(E(1), di);

if (~(img(imag(Pe),real(Pe))) && (img(imag(Pi),real(Pi))))
bol = 1;
else
```

```
dpc = dpc + 1;
end
end
```

2.A. Algoritmo para extrair o Contorno
Fonte: EP7 - Algoritmo do Ceguinho e calculo da curvatura multi-escala . Análise e Reconhecimento de Formas: Teoria e Prática. Prof: Roberto Marcondes César Junior e o aluno Milton YutakaNishiyamaJunior .

```
function [Maxx,Minn] =extract_contour(semruido,E)
figure,hold;
colormap(gray(256));
title('(contorno)')

image(semruido * 128);

% Calcula a curvatura e desenha o morfograma
figure;
hold;

for t = 1:32
    [kauxrf] = curvogram(E, t);
plot(rf);

    % Copia a curvatura para uma matriz
fori = 1 : max(size(kaux))
k(i, t) = kaux(i);
end

end
title('(desenha o morfograma)')

% plota o curvograma
figure,
mesh(k);
hold;
xlabel('t = 1/a');
ylabel('t');
zlabel('k');
title('Curvograma');

[Maxx,Minn]=picosvalas2(k);
```

3.A Curvograma
Fonte: EP7 - Algoritmo do Ceguinho e calculo da curvatura multi-escala . Análise e Reconhecimento de Formas: Teoria e Prática. Prof: Roberto Marcondes César Junior e o aluno Milton YutakaNishiyamaJunior .

```
function [k, rf] = curvogram(E,t)
F = fftshift(fft(E));
M = max(size(F));

% Escala da freqüência
```

```
N = max(size(E));
T = 2 * pi / (N - 1);

n = 1;

a = (-floor(N / 2));
b = (N - floor(N / 2) - 1);

%calculando a frequencia
for s = a:b
f(n) = s / (N * T);
n = n + 1;
end;

% Transformada da 1a derivada
for i = 1: M
dF(i) = 2 * pi * f(i) * 1j * F(i);
end

% Transformada da 2a derivada
fori = 1: M
d2F(i) = (2 * pi * f(i) * 1j) ^ 2 * F(i);
end

% Aplica o filtro da gaussiana em U, dU e ddU
for i = 1 : M
    Ga(i) = exp(-(2 * pi) ^ 2 * f(i) ^ 2 / (2 * t ^ 2));
    F(i) = F(i) * Ga(i);
dF(i) = dF(i) * Ga(i);
d2F(i) = d2F(i) * Ga(i);
end

rf = ifft(unshift(F));% fazendo a reconstrução da curva depois de aplicar
%o filtro da Gaussiana

% derivadas estimadas
df = ifft(unshift(dF));
d2f = ifft(unshift(d2F));

% constante de normalização do efeito de "shrinking"
NC = perim(E) / perim(rf);

% fazendo a normalização
for i = 1 : N
df(i) = df(i) * NC;
d2f(i) = d2f(i) * NC;
end

% Cálculo da curvatura
for i = 1 : N
k(i) = -imag(df(i) * conj(d2f(i))) / abs(df(i)) ^ 3;
end

%subrotinas
function r = perim(c)

r = 0;
```

```
fori = 2 : max(size(c));
r = r + abs(c(i) - c(i - 1));
end



function b = unshift(a)

L = max(size(a));
shift = floor(L / 2);

fori = 1 : (L - shift)
b(i) = a(i + shift);
end

fori = 1: shift
b(i + (L - shift)) = a(i);
end
```

APÊNDICE

A.1 Algoritmo de Binarização

```
function B = FBinarizacao(I)
Y = rgb2gray(I);            %acha o branco e preto puro
Y=255-Y;                    %muda pra negativo   para funcionar para fundo
                            %branco ou claro senão tira essa linha
T = graythresh(Y);          %Imagem em escala cinza usando método de Otsu.
B = im2bw(Y,T);             %Binarizando imagem

end
```

A.2 Algoritmo de Limpeza de ruídos

```
function [A,AreaFolha] = LimpezaRuidos(B)
[L,num] = bwlabel(B,8);     %retorna uma matriz L, do mesmo tamanho como
                             %BW, contendo rótulos para os objetos
                             %conectados em BW.

%Inicializando variáveis
maiorid = 0;                 %armazena maior indice da classificação
somav = 0;                   %vetor maior somado

[a,b] = size(L);

for i=1:1:num,
  [l,c,ve] = find(L==i);     %[linha,coluna,vetor] = find(matrix);
soma = sum(ve(:));          %v é passado para vetor coluna

if soma >somav,
somav = soma ;          %soma a área
maiorid = i ;
end
end

[l,c,ve] = find(L==maiorid);
AreaFolha=sum(ve(:));

A=L;
fori=1:1:a,
for j=1:1:b,
if L(i,j) ~= maiorid     %~= é o sinal de diferente
A(i,j)= 0;
end
end
end
end
```

A.3 Algoritmo de Momentos

```
functionMo = Momentos(B) %imagem binarizada
Dados = double(B);
[l,c] = size(Dados);
N = max(size(Dados));
Media = mean(Dados);            %Momentos de Ordem 1
Variancia = var(Dados);         %Momentos de Ordem 2
for j=1:1:N,
Mo(1,j) = 0;
```

```
end

fori=1:1:l,
for j=1:1:c,
Mo(1,i) = Mo(1,i) +((Dados(i,j) - Media(i))^3);
end
Mo(1,i) = Mo(1,i)/(N-1);
end
```

A.4 Picos e Valas

```
function [Maxx,Minn] =picosvalas2(k) %vai ter que retornar o guarda max e
guarda min
%k=rand(40,2);

[tpont,ncurv]=size(k)
maximo=k(1,1);
minimo=k(1,1);
cont=1;
cont1=1;

maxatual=max(k(2,:));
 maxatual1=max(k(1,:));

controle=0;
ifmaxatual>maxatual1,
controle=1; %controle 0 pra max 1 para min
end

%inicializavariaveis
indxx=1;
indyy=min(k(1,:));
indx=1;
indy=max(k(1,:));
fori = 1 : tpont
maxatual=max(k(i,:));
minatual=min(k(i,:));
ifmaxatual>maximo,
maximo=maxatual;
indx=i;
indy=maxatual;
cont=cont+1;
end
ifmaxatual<maximo,
maximo=maxatual;
guardamax(cont,1)=indx;
guardamax(cont,2)=indy;



end

ifminatual<minimo,
minimo=minatual;
indxx=i;
indyy=minatual;
cont1=cont1+1;
end
ifminatual>minimo,
```

```
minimo=minatual;
guardamin(cont1,1)=indxx;
guardamin(cont1,2)=indyy;

end
end

Maxx=guardamax;
Minn=guardamin;

figure,
hold on %para sobreporplotagem
plot(k,'--rs','MarkerFaceColor','y','MarkerSize',2)
plot(guardamin(:,1),guardamin(:,2),'--
rs','MarkerFaceColor','m','MarkerSize',4)

%title('cava')
%figure,
plot(guardamax(:,1),guardamax(:,2),'--
rs','MarkerFaceColor','g','MarkerSize',4)
%axis([0 tpont -1.8 0.2]) para delimitar o plot
title('pico')
```

A.5 PCA (TKL)

```
functionFinalDataAdjust = PCA(borda)
% começar PCA

Dados=borda;
Media=mean(Dados);

DataAdjust(:,1)= Dados(:,1) - Media(1);  %x - mediax  todos da coluna 1
menos a media 1
DataAdjust(:,2)= Dados(:,2) - Media(2); %y - mediay  todos da coluna 2
menos a media 2

Mcovariancia=cov(Dados); %calculo da matriz de covariancia
Mcovariancia
%calculo dos autovetores e autovalores da matriz de covariancia

[v,lambda]=eig(Mcovariancia);  % v*lambda = K

%Mcovariancia
%v       %autovetor
%lambda  %autovalor
lamb=diag(lambda); %pega os valores do autovalor

[lb,ind]=sort(lamb,1,'descend');

%toma os autovetores de acordo com a ordem decrescente dos autovalores
NewVet=v(:,ind); %Cada vetor é um vetor coluna, mx1
                 %Os vetores são os novos eixos, perpendiculares entre si
%calculo dos eixos para plotagem,

rmax=[Media-200*(NewVet(:,1)).'; Media+200*(NewVet(:,1)).']; %u é a média
rmin=[Media-200*(NewVet(:,2)).'; Media+200*(NewVet(:,2)).'];
```

```
%*************************************************************************
%Rotacionando dados ajustados
%*************************************************************************
RowFeature=NewVet.';%Matriz transposta dos autovetores para PCA
                    %         |    |                      -V1-
                    %Antes   V1   V2  transposto ==>
                    %         |    |                      -V2-
RowDataAdjust=DataAdjust.';%Matriz dados ajustados para PCA. De mxn para
nxm
FinalDataAdjust=(RowFeature*RowDataAdjust).';%Dados ajustados rotacionados
%  -V1-      -X-
                                              %        *
%  -V2-      -y-

%AutoVet(n,n)' x Dados(m,n)' = NovosDados(n,m)'=DadoRotacionados(m,n)
%*************************************************************************
%Impressao dos dados brutos
%*************************************************************************

figure,
subplot(2,2,1);
plot(Dados(:,1),Dados(:,2),'k.',rmax(:,1),rmax(:,2),rmin(:,1),rmin(:,2),'k-
')
title('dados')



rmax=[-200*(NewVet(:,1)).'; 200*(NewVet(:,1)).']; %u é a média
rmin=[-200*(NewVet(:,2)).'; 200*(NewVet(:,2)).'];
rmax=[-300 0; 300 0];%//eixos x e y normais
rmin=[-0 -250; 0 250];
subplot(2,2,2);
plot(DataAdjust(:,1),DataAdjust(:,2),'k.',rmax(:,1),rmax(:,2),rmin(:,1),rmi
n(:,2),'k-')
title('Dados ajustados (media zero)')

FinalData=((NewVet.')*(Dados.')).';%//DadosRotacionados=(Autovet.'*Dados.')
.
udat=mean(FinalData);
xdat=[-400 udat(2);200 udat(2)];
ydat=[udat(1) -100;udat(1) -500];

subplot(2,2,3);
plot(FinalData(:,1),FinalData(:,2),'k.',xdat(:,1),xdat(:,2),ydat(:,1),ydat(
:,2))
title('DadosRotacionados=([Autovetores]T.[Dados]T)T')

rmax=[-300 0; 250 0];%//eixos x e y normais
rmin=[-0 -200; 0 200];

subplot(2,2,4);
plot(FinalDataAdjust(:,1),FinalDataAdjust(:,2),'k.',rmax(:,1),rmax(:,2),rmi
n(:,1),rmin(:,2))
title('Dados media zero rotacionados pelo maior eixo (grosso)')
end
```

A.6 Centroide

```
function [x,y] = centroide(tima)%imagem binarizada
x=sum(tima(:,1));
```

```
y=sum(tima(:,2));
[area,v]=size(tima);
 x=x/area;
 y=y/area;

figure,
hold on,
plot(x,y,'--rs','MarkerSize',5)
title('centroide')

plot(tima(:,1),tima(:,2),'r');
title('centro')
end
```

A.7Diâmetro

```
function ID = Diametro(ImB)
[n,a] = size(ImB); %coordenadas  do perimetro

dmax=0;
for i=1:1:(n-1),
for j=(i+1):1:n,
d=DistamciaPontos(ImB(i,1),ImB(i,2),ImB(j,1),ImB(j,2));
if(d>dmax)
dmax=d;
ID(1,1)=ImB(i,1);
ID(1,2)=ImB(i,2);
ID(2,1)=ImB(j,1);
ID(2,2)=ImB(j,2);
end
end
end
ID
%plot(IDX,IDY,'k.',IDX1,IDY2)

plot(ID(:,1),ID(:,2),'--rs','LineWidth',2)

%plot(IDX1,IDY2,'--rs','LineWidth',8)

plot(ImB(:,1),ImB(:,2), 'black', 'LineWidth', 2)

end
```

A.8 Distância entre dois pontos

```
function d = DistamciaPontos(x,y,x1,y1)  %distancia entre dois pontos
d = sqrt(((x-x1)^2)+((y-y1)^2));
```

A.9 Distânciamedia até a borda

```
function DM = DistanciaMediaBorda(xs,ys,X,Y)  %entrada com função
conplexo(E) saida com distancia media
%distancia media da borda até o centro da figura que é o zero ou o centro
de massa

DM=0;

%fazer um for colocando dotos as coordenadas do perimetro
```

```
[b,a]=size(xs);



fori=1:1:b,
DM = DistamciaPontos(xs(i),ys(i),X,Y)+DM;
end

DM=DM/b;
end
```

A.10Simetria

```
function []= simetria(element) %elemente tem que estar preenchido no caso
não pode ser preenchido

aa=max(max(element(:,2)));

[l,c,v]=find([element(:,1),element(:,2)]==[element(:,1),aa-element(:,2)-
aa]);
[l,c,v2]=find([element(:,1),element(:,2)]~=[element(:,1),aa-element(:,2)-
aa]);

%simetria=v/v2;
figure,
hold on,
plot(element(:,1),element(:,2),'r')
plot(element(:,1),(aa-element(:,2)-aa),'y')

sime=figure;
hold on,
fill(element(:,1),element(:,2),'k','FaceAlpha', 0.2)
fill(element(:,1),(aa-element(:,2)-aa),'r','FaceAlpha', 0.5)
title('simetria')
saveas(sime,'C:\Users\Priscila\Documents\tcc\simetria','jpg')

simetria1=figure;
A = fill(element(:,1),element(:,2),'k');
axis image off  %imagemsemoseixos x e y
saveas(simetria1, 'arquivo','png')   %local para salvar a imagem
I = imread('arquivo','png');
B = im2bw(I);
C = flipud(B);   %invertendo imagem horizontalmente

[a,t]=size(B);
p = 0;
fi = 0;



for i=1:1:a,    %achando os pixels de intersecção
for j=1:1:t,
if (B(i,j) == 0) && (C(i,j) == 0)
D(i,j) = 0;
            p = p+1;
fi = fi+1;
else
D(i,j) = 1;
            %p = p+1;
if (B(i,j) == 0)
```

```
fi = fi+1;
end
end
end
end

imshow(D)
title('Interseccao - Simetria')

Simetria = p/fi;
Simetria
end
```

A.11 Algoritmo do Programa Principal

```
%Criando arquivo que guardará os picos
fid2 = fopen('arquivo','wt');
I = imread('arquivo.png');

%Plotando folha
subplot(2,2,1);
imshow(I);
title('original')

%Realizando a Binarização
B=FBinarizacao(I);          %-------------------------

%Folha binarizada
subplot(2,2,2);
imshow(B);
title('Binarizada')

%Realizando a limpeza de ruídos
[semruido,AreaFolha]=LimpezaRuidos(B);

%Plot - Limpeza de ruídos
subplot(2,2,3);
imshow(semruido);
title('limpesa de ruidos')

Mo = Momentos(semruido);  %Cálculo de Momentos
Media_Mo = mean2(Mo);     %Cálculo da média de Momentos
E=contour(semruido);
perimetroo=Perimetro(E);
compacidade=((perimetroo*perimetroo)/AreaFolha);

%Plot de somente da borda da folha
subplot(2,2,4);
plot(E,'r');
title('(borda)')

figure,
area(semruido);

[Maxx,Minn]=extract_contour(semruido,E);

[a,t]=size(Maxx);
```

```
fprintf(fid2,'Picos: \n');
fori = 1 : a
fprintf(fid2,'%f   %f\n',Maxx(i,1),Maxx(i,2));
end
MediaPico=mean(Maxx(:,2));
```

A.12 Principal Component Analysis (PCA)

```
% Verificando correlação entre as variáveis
correlacao = corr(dados,dados);

[wcoeff,score,latent,tsquared,explained] = pca(dados,...
'VariableWeights','variance');

cumsum(latent)./sum(latent)
%3 coeficientes principais
c3 = wcoeff(:,1:3)

%Transformando componentes
coefforth = inv(diag(std(dados)))*wcoeff;

%Verificando ortogonalidade
I = c3'*c3

%calculando scores
cscores = zscore(dados)*coefforth;

figure()
plot(score(:,1),score(:,2),'o')
xlabel('1st Principal Component')
ylabel('2nd Principal Component')

%Gráfico que mostra a porcentagem de expicação de cada componente principal
figure()
pareto(explained)
xlabel('Principal Component')
ylabel('Variance Explained (%)')
```

Arquivo dados_com_pca.arff

```
@RELATION folha

@ATTRIBUTE variavel1  NUMERIC
@ATTRIBUTE variavel2  NUMERIC
@ATTRIBUTE variavel3  NUMERIC
@ATTRIBUTE variavel4  NUMERIC
@ATTRIBUTE class{Folha-pitanga,Folha-coquecasa,Folha-
jequetiba,Folha-lima,Folha-limao,Folha-desc1,Folha-desc2,Folha-
desc3,Folha-desc4,Folha-desc5}
@DATA
1.8315,-1.1439,2.4944,-0.1135,Folha-pitanga
1.6216,-0.5289,1.6099,-0.2400,Folha-pitanga
0.8795,0.4729,0.9887,-0.1063,Folha-pitanga
0.8932,0.9118,0.5383,-0.1641,Folha-pitanga
```

```
2.2474,-0.6270,1.3405,-0.2117,Folha-pitanga
2.3209,-1.1819,1.8219,-0.1006,Folha-pitanga
1.3711,0.1274,1.1444,-0.1144,Folha-pitanga
1.5349,-0.2318,1.5616,-0.0758,Folha-pitanga
2.0511,-0.3443,0.6974,-0.0605,Folha-pitanga
2.2123,0.0865,0.2577,-0.1488,Folha-pitanga
2.7839,-1.1353,0.8310,-0.1480,Folha-pitanga
2.7302,-1.1614,0.7220,-0.1478,Folha-pitanga
1.9310,-0.1411,0.8717,-0.0838,Folha-pitanga
2.2351,-0.2155,1.2661,-0.0497,Folha-pitanga
1.9796,0.1339,0.3049,-0.1123,Folha-pitanga
2.1013,-0.7310,1.1659,0.0129,Folha-pitanga
1.9796,0.1339,0.3049,-0.1123,Folha-pitanga
1.9796,0.1339,0.3049,-0.1123,Folha-pitanga
2.2363,-0.5958,1.3016,-0.1312,Folha-pitanga
2.2595,-0.5735,1.2571,-0.1741,Folha-pitanga
1.6435,-0.8219,1.8727,-0.0011,Folha-pitanga
1.6292,-0.7500,1.8246,-0.0121,Folha-pitanga
1.2827,-0.6282,2.5144,0.0278,Folha-pitanga
1.6403,-1.4067,3.5710,0.1774,Folha-pitanga
2.5170,-1.1423,1.0734,-0.2270,Folha-pitanga
2.9270,-1.5197,1.7893,-0.1262,Folha-pitanga
-2.8321,-1.5552,-0.0581,0.0389,Folha-coquecasa
-2.9883,-1.8347,-0.0358,0.1204,Folha-coquecasa
-3.1671,-4.3128,-7.8177,-10.0906,Folha-coquecasa
-2.1833,-2.4926,-0.6115,0.1134,Folha-coquecasa
-3.1030,-2.3897,-0.6144,0.1819,Folha-coquecasa
-2.2157,-2.2315,-0.0095,0.1498,Folha-coquecasa
-2.9428,-2.7847,0.0672,0.3435,Folha-coquecasa
-2.1568,-2.4188,0.6125,0.4184,Folha-coquecasa
-2.4992,-2.4059,1.5924,0.4048,Folha-coquecasa
-4.0323,-2.0569,-0.5084,0.1417,Folha-coquecasa
-3.5764,-0.7461,-0.5271,0.1310,Folha-coquecasa
-3.9629,-1.7334,-0.4216,0.1046,Folha-coquecasa
-3.8228,-1.0676,2.0476,0.3287,Folha-coquecasa
-4.9663,-0.4433,0.1942,0.1019,Folha-coquecasa
-2.3314,-1.7575,0.4783,0.1421,Folha-coquecasa
-2.4962,-1.7977,0.3513,0.2291,Folha-coquecasa
-2.9277,-1.3699,0.7153,0.1208,Folha-coquecasa
-3.5713,-1.0105,-0.1613,0.1293,Folha-coquecasa
-2.3154,-3.0510,0.3000,0.2820,Folha-coquecasa
-2.0272,-2.8044,0.2626,0.1655,Folha-coquecasa
-3.4649,-1.1855,0.7780,0.2428,Folha-coquecasa
-3.5633,-1.6978,1.0363,0.2624,Folha-coquecasa
-3.4360,-1.5295,0.9930,0.1736,Folha-coquecasa
-3.9605,-1.9206,0.8717,0.2726,Folha-coquecasa
-3.2336,-2.6801,-0.1168,0.1228,Folha-coquecasa
-2.9616,-2.5407,-0.0381,0.0430,Folha-coquecasa
-2.1162,-3.4343,-0.4592,0.1349,Folha-coquecasa
-1.8147,-3.4007,-0.4144,0.0891,Folha-coquecasa
-2.2769,-2.8363,0.6508,0.4126,Folha-coquecasa
-2.8199,-3.1949,0.2377,0.3297,Folha-coquecasa
-2.9229,-1.6428,0.1120,0.0394,Folha-coquecasa
-3.2160,-0.8337,-0.9129,-0.1047,Folha-coquecasa
```

```
-3.9640,-0.9620,0.2305,0.1994,Folha-coquecasa
-3.7407,-0.4091,-0.1089,0.1807,Folha-coquecasa
-2.8829,-1.1245,-0.5063,0.0329,Folha-coquecasa
-2.7051,-2.3040,0.4989,0.2171,Folha-coquecasa
-3.2496,-2.6396,-0.3334,0.2704,Folha-coquecasa
-2.5328,-2.6287,-0.0365,0.0763,Folha-coquecasa
-4.1262,-0.2507,0.3217,0.1734,Folha-coquecasa
-3.8989,-1.6472,1.4932,0.2482,Folha-coquecasa
-1.3595,-3.4586,-1.1428,-0.0136,Folha-coquecasa
-1.3554,-3.3389,-1.0406,0.0511,Folha-coquecasa
-3.8278,-1.9525,1.2088,0.5355,Folha-coquecasa
-4.5940,-2.0384,0.1997,0.3103,Folha-coquecasa
1.3948,0.5830,-1.3737,-0.0140,Folha-jequetiba
1.6444,-0.0309,-0.5998,0.0751,Folha-jequetiba
1.8342,-0.0783,-0.7807,0.0296,Folha-jequetiba
2.2007,-0.5121,-0.0560,0.1677,Folha-jequetiba
1.3386,0.3401,-0.7735,0.0844,Folha-jequetiba
1.8655,0.3621,-0.4402,-0.0418,Folha-jequetiba
1.3533,0.8762,-1.4998,-0.0498,Folha-jequetiba
1.3531,-0.3893,-0.4934,0.1809,Folha-jequetiba
1.8748,0.4337,-0.8366,-0.0797,Folha-jequetiba
1.7696,1.0310,-1.6175,-0.2143,Folha-jequetiba
1.4567,1.1743,-1.3882,-0.2019,Folha-jequetiba
1.3889,0.7568,-1.2106,-0.1122,Folha-jequetiba
1.3596,0.3209,-1.2201,-0.0814,Folha-jequetiba
1.6815,-0.2283,-0.4898,0.0559,Folha-jequetiba
2.5770,-0.1337,-1.4171,-0.0824,Folha-jequetiba
2.4677,0.1371,-1.9400,-0.2416,Folha-jequetiba
0.9490,1.0826,-1.5734,-0.0495,Folha-jequetiba
1.1716,1.0393,-1.2802,-0.0069,Folha-jequetiba
3.0695,-1.5286,0.5483,-0.2179,Folha-lima
2.9641,-1.1775,-0.0027,-0.3237,Folha-lima
1.0011,0.2743,0.6660,0.0005,Folha-lima
0.9438,1.0747,-0.0837,-0.1308,Folha-lima
1.9444,0.5549,-0.5463,-0.4377,Folha-lima
1.7602,0.7449,-0.8677,-0.4201,Folha-lima
2.0073,-0.5714,0.5313,-0.1664,Folha-lima
1.9745,-0.2544,0.2282,-0.2478,Folha-lima
1.0557,0.5655,-0.0018,0.0193,Folha-lima
1.1862,0.7660,-0.0394,0.0157,Folha-lima
1.0743,0.8435,-0.1279,-0.0838,Folha-lima
1.2727,0.7450,-0.0223,-0.1627,Folha-lima
3.4090,-1.6767,-0.0683,-0.3463,Folha-lima
3.2027,-1.3600,-0.5926,-0.4313,Folha-lima
3.3480,-1.6918,-0.3053,-0.4919,Folha-lima
3.5286,-1.2927,-0.5251,-0.5627,Folha-lima
1.9980,-0.5632,0.6121,-0.2023,Folha-lima
2.2649,0.0821,-0.6615,-0.5309,Folha-lima
2.3969,-0.7050,0.3083,-0.3167,Folha-lima
0.6984,1.2274,0.5083,-0.0541,Folha-limao
0.7284,0.9814,0.8874,0.0529,Folha-limao
3.2707,-1.6122,-0.0087,-0.2730,Folha-limao
3.2313,-1.3284,-0.1167,-0.2628,Folha-limao
1.8295,0.7199,-0.5896,-0.2438,Folha-limao
```

```
1.7768,0.6147,-0.4738,-0.2590,Folha-limao
2.3592,-1.0371,0.4003,-0.2499,Folha-limao
2.5488,-0.5675,0.3187,-0.3378,Folha-limao
2.2249,-0.6703,0.3413,-0.1477,Folha-limao
2.3670,-0.6232,0.5447,-0.1345,Folha-limao
3.1195,-2.1232,1.0539,-0.0892,Folha-limao
2.8781,-1.4742,0.5287,-0.1380,Folha-limao
2.6886,-1.1265,0.1208,-0.2574,Folha-limao
2.6768,-1.3933,0.4026,-0.1527,Folha-limao
0.9507,1.0641,0.1734,-0.2417,Folha-limao
0.7949,1.2569,0.1311,-0.2124,Folha-limao
3.6563,-1.9310,0.1121,-0.3453,Folha-limao
3.3956,-1.9040,-0.0297,-0.2642,Folha-limao
0.7796,1.9299,0.3597,-0.2461,Folha-limao
0.5725,1.0362,1.1277,-0.0041,Folha-limao
-1.3175,1.8361,0.0229,0.2619,Folha-desc1
-0.9948,1.6640,-0.1394,0.1391,Folha-desc1
1.0490,0.4460,-1.1539,0.1169,Folha-desc1
0.8250,0.3684,-1.4652,0.1811,Folha-desc1
0.8036,0.5129,-0.5873,0.2359,Folha-desc1
1.1338,0.1481,-1.0264,0.1357,Folha-desc1
-0.0714,1.3957,0.3639,0.1759,Folha-desc1
-0.3863,1.4896,-0.1979,0.2061,Folha-desc1
-0.2677,1.3278,-1.7879,0.1744,Folha-desc1
0.8557,0.3803,-0.2350,0.2209,Folha-desc1
-1.1470,1.1846,-0.5850,0.3833,Folha-desc1
-0.5070,1.0952,-0.4301,0.2751,Folha-desc1
-0.4462,1.1364,0.4788,0.2597,Folha-desc1
-0.8781,1.0883,0.1451,0.3273,Folha-desc1
0.6480,0.5513,-0.6857,0.1694,Folha-desc1
-0.0343,0.5001,-1.1077,0.2741,Folha-desc1
0.4339,1.5647,-1.0228,0.0427,Folha-desc1
0.4596,0.9880,-0.6482,0.1562,Folha-desc1
-0.5315,1.4041,0.1215,0.1694,Folha-desc1
-0.0554,1.3117,0.5709,0.1752,Folha-desc1
0.5655,0.3919,-0.3126,0.2273,Folha-desc1
0.9407,0.8157,-0.7025,0.0845,Folha-desc1
0.4424,0.6905,-0.3597,0.1783,Folha-desc1
0.6062,0.9785,-0.3712,0.2032,Folha-desc1
-1.4273,1.4978,0.1075,0.3568,Folha-desc1
-1.4524,0.9943,0.2339,0.3601,Folha-desc1
0.7998,0.8121,0.0455,0.1478,Folha-desc1
0.8856,0.2671,0.5545,0.2629,Folha-desc1
-2.7978,4.7358,7.0781,-11.3858,Folha-desc2
-2.3887,3.1251,0.4292,0.2263,Folha-desc2
-2.9199,3.1849,-0.0063,0.3618,Folha-desc2
-3.1717,3.9111,-1.0530,0.1799,Folha-desc2
-1.2067,2.0993,0.0260,0.2317,Folha-desc2
-1.3302,2.0187,0.0388,0.2679,Folha-desc2
0.6170,0.8502,0.3490,0.1562,Folha-desc2
0.1658,1.7236,-0.7262,0.0953,Folha-desc2
-0.6821,2.2486,-1.1222,0.1920,Folha-desc2
-0.0143,0.9600,0.1348,0.1789,Folha-desc2
1.3478,-1.1902,-0.8506,0.1349,Folha-desc2
```

```
1.2636,-1.0156,-0.8980,0.1328,Folha-desc2
-0.6662,1.9659,-1.2327,0.1877,Folha-desc2
-0.7534,2.3705,-1.8450,0.0282,Folha-desc2
-2.4725,3.0305,0.8144,0.2106,Folha-desc2
-2.4514,2.7751,0.8126,0.1776,Folha-desc2
-2.5576,3.2113,-0.4021,0.1698,Folha-desc2
-2.5795,3.2759,-0.6068,0.0981,Folha-desc2
-1.9222,2.2454,-0.3500,0.1944,Folha-desc2
-2.0746,2.7241,-0.9873,0.1092,Folha-desc2
-1.9355,2.1767,-0.1179,0.3009,Folha-desc3
-2.2484,2.1252,-0.2928,0.2964,Folha-desc3
0.7051,-0.1220,0.3889,0.2046,Folha-desc3
0.7019,0.5233,-0.0962,0.1202,Folha-desc3
0.9056,0.2620,-1.1066,0.2312,Folha-desc3
0.7110,0.5488,-1.0586,0.3641,Folha-desc3
0.8829,-0.4129,-0.0988,0.3897,Folha-desc3
1.0023,0.0200,-0.4303,0.2892,Folha-desc3
-0.3712,0.1437,1.1530,0.4021,Folha-desc3
-0.1718,0.8344,0.7925,0.2359,Folha-desc3
-1.7788,1.1521,0.7104,0.7432,Folha-desc3
-2.8196,2.3307,-0.9994,0.6631,Folha-desc3
-0.6225,0.1740,0.0622,0.6165,Folha-desc3
-0.2690,0.9075,-0.4942,0.4100,Folha-desc3
-0.8249,1.2002,-0.1884,0.2234,Folha-desc3
-0.8689,1.4087,-0.2490,0.3431,Folha-desc3
-0.9429,1.9660,-1.0197,0.0509,Folha-desc3
-0.6551,0.7733,0.4263,0.3587,Folha-desc3
-1.2763,0.6162,0.7762,0.2421,Folha-desc4
-1.2667,1.0853,0.4680,0.1976,Folha-desc4
-1.1646,1.1292,0.0607,0.1907,Folha-desc4
-1.4429,1.2298,0.0086,0.2094,Folha-desc4
-1.4532,1.2576,0.4791,0.2071,Folha-desc4
-1.2692,1.0642,0.8754,0.2729,Folha-desc4
-0.8298,1.3124,0.5278,0.2187,Folha-desc4
-1.4523,1.8452,-0.4817,0.1477,Folha-desc4
-1.7596,1.6550,1.0475,0.2401,Folha-desc4
-1.5811,2.2875,0.6778,0.1246,Folha-desc4
-0.0422,1.1418,-0.8749,0.0952,Folha-desc4
0.6360,0.9514,-0.2522,0.0574,Folha-desc4
0.1105,0.6397,-0.1476,0.1878,Folha-desc4
0.3521,0.2312,0.1920,0.1540,Folha-desc4
1.1897,0.5628,-0.9576,-0.0263,Folha-desc4
1.3490,0.4448,-0.6475,0.0097,Folha-desc4
0.6342,0.5046,-0.7149,0.1669,Folha-desc4
0.6858,0.1142,-0.2829,0.1964,Folha-desc4
-0.9847,1.7215,-0.0159,0.1261,Folha-desc4
-0.8545,1.0865,0.7352,0.2689,Folha-desc4
-0.1089,1.4807,-0.5162,0.0770,Folha-desc4
0.2786,0.6611,0.5481,0.2166,Folha-desc4
-0.2758,0.8193,0.4391,0.2846,Folha-desc4
-0.3171,0.5762,0.5479,0.3247,Folha-desc4
1.4678,-0.3675,-0.0717,0.2063,Folha-desc4
1.2224,-0.1556,-0.5421,0.1240,Folha-desc4
-0.0057,1.1969,-0.6595,0.1180,Folha-desc4
```

```
0.1965,0.0949,0.4748,0.3482,Folha-desc4
-0.3649,1.6052,-1.5107,-0.0308,Folha-desc4
-0.5099,0.7332,-0.5357,0.2391,Folha-desc4
-1.4130,2.3678,0.1158,0.1835,Folha-desc4
-1.3137,1.9877,0.3408,0.1371,Folha-desc4
0.2522,-0.2188,0.0772,0.2260,Folha-desc5
0.5220,0.1732,-0.0251,0.2105,Folha-desc5
-0.1006,0.0064,-1.1990,0.3167,Folha-desc5
0.9489,-0.4100,-0.4029,0.2336,Folha-desc5
0.7494,-0.9789,-0.2491,0.1463,Folha-desc5
0.9268,-0.2407,-0.7217,0.0508,Folha-desc5
0.5049,-0.0294,-0.2074,0.2512,Folha-desc5
0.6220,-0.1864,-0.4206,0.2050,Folha-desc5
0.4634,0.2701,-0.6816,0.1264,Folha-desc5
1.1604,-0.5197,-1.0390,0.0881,Folha-desc5
1.4703,-1.2635,-0.4048,0.1496,Folha-desc5
1.4610,-1.1877,-0.4471,0.1992,Folha-desc5
1.4436,-1.6490,-0.4572,0.1157,Folha-desc5
0.1501,0.3921,-0.0094,0.1638,Folha-desc5
0.4782,-0.8019,1.0447,0.3244,Folha-desc5
-0.0285,0.0645,-0.1680,0.1564,Folha-desc5
0.7364,-0.3342,0.8762,0.2258,Folha-desc5
-0.1158,0.3936,-0.2344,0.1625,Folha-desc5
0.2614,0.1567,0.3313,0.1874,Folha-desc5
1.2924,-1.9194,-0.5922,0.2328,Folha-desc5
1.8745,-1.7326,-0.3066,0.1759,Folha-desc5
0.3574,0.3640,-1.1484,0.1425,Folha-desc5
1.0874,-1.1708,0.8116,0.4808,Folha-desc5
```