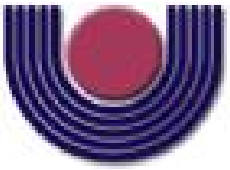

Unioeste - Universidade Estadual do Oeste do Paraná
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
Colegiado de Ciência da Computação
Curso de Bacharelado em Ciência da Computação

Extração de Características de Imagens de Morfotipos de Fitólitos da Familia Arecaceae

Luiz Gustavo de Souza

CASCAVEL
2012

LUIZ GUSTAVO DE SOUZA

EXTRAÇÃO DE CARACTERÍSTICAS DE IMAGENS DE MORFOTIPOS DE FITÓLITOS DA FAMILIA ARECACEAE

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação, do Centro de Ciências Exatas e Tecnológicas da Universidade Estadual do Oeste do Paraná - Campus de Cascavel

Orientador: Prof. Dr. Clodis Boscarioli

CASCAVEL
2012

LUIZ GUSTAVO DE SOUZA

EXTRAÇÃO DE CARACTERÍSTICAS DE IMAGENS DE MORFOTIPOS DE FITÓLITOS DA FAMILIA ARECACEAE

Monografia apresentada como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação, pela Universidade Estadual do Oeste do Paraná, Campus de Cascavel, aprovada pela Comissão formada pelos professores:

Prof. Dr. Clodis Boscarioli (Orientador)
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Msc. André Luiz Brun
Colegiado de Ciência da Computação,
UNIOESTE

Profª. Dra. Márcia Regina Calegari
Colegiado de Geografia, UNIOESTE

Cascavel, 7 de dezembro de 2012

“I’ll always remember, those were the best of times
I’ll cherish them forever, the best of times”.
(The Best of Times, Dream Theater)

DEDICATÓRIA

A Luiz Cláudio, o pai que abriu as portas para todas as oportunidades. A Clodis Boscarioli, quem proporcionou inúmeras oportunidades e ensinamentos. A Marília, que suportou os momentos mais difíceis deste trajeto ao meu lado.

AGRADECIMENTOS

Primeiramente a meu pai, Luiz Cláudio de Souza, por todo o apoio e ensinamento, que foram fundamentais para meu crescimento e vitórias. Também a minha mãe, Sirley Aparecida Barbetto, pelo apoio e compreensão, que apesar de todos os obstáculos vividos conseguiu se fazer presente.

À toda a minha família, que sempre me apoiou e se preocupou comigo mesmo com a distância, proporcionando-me o incentivo necessário. E, principalmente, à minha avó Lourdes, que sempre será a mãe que me criou desde a infância, e ainda hoje mantém sua posição com muitos mimos.

À minha namorada Marília, que deu o maior apoio para minha formação, e aguentou comigo todo a carga de trabalho dos últimos anos e a minha falta de tempo.

À minha segunda família, Pastel, Ale, Baiano, Bobs, Greg e Tatá, que me proporcionaram ótimos momentos e aprendizados, que levarei para toda minha vida. Jamais esquecerei todos os bons momentos que passamos, todas as dificuldades e lutas que fizeram de nós verdadeiros irmãos.

Aos grandes amigos Fernando Spanhol, o "Span"e Vini, que de colegas de turma passaram a grandes amigos sempre presentes, apoiando e principalmente contribuindo em trabalhos e provas por várias noites, e claro, muitos jogos. Também a Juliano Richetti, grande parceiro em trabalhos.

A todos os amigos e colegas que conheci durante esta trajetória, que muito apoiaram e contribuíram com suas experiências de vida, sempre lembrarei das festas e momentos divertidos.

Ao Professor Doutor Ex-Diretor Tutor Orientador Clodis Boscarioli, que com todos os puxões de orelha me fez chegar a este ponto, e pronto para novas batalhas e desafios. Será sempre lembrado por mim como a pessoa boa e justa que influencia e incentiva diretamente todos a sua volta.

Aos professores André, Adriana, Adair, Márcio e Edmar, os grandes professores que me marcaram com a boa conduta, apoio e interesse em sempre ajudar, independente da situação.

Aos demais professores, que mantém este Curso com a excelência que reflete sobre as notas em exames e competições nacionais.

Aos grandes e virtuosos Mike Portnoy, John Petrucci, Michael Romeo, Russell Allen, Tobias Sammet, James Hetfield, Rafael Bitencourt, Eduardo Ardanuy, Andria e Ivan Busic, André Matos e as bandas Symphony X, Dream Theater, Flying Colors, Metallica, Angra, Dr. Sin, Avantasia, Edguy, Seventh Avenue, Oficina G3 e todas as outras bandas que mantém vivos o Rock and Roll, Progressivo e o Metal.

Lista de Figuras

Lista de Algoritmos

Sumário

Resumo

Extração de características é um processo que visa a obtenção de informações quantitativas a partir de imagens digitais, que sejam capazes de diferenciá-las, convertendo cada tipo de imagem em dados para reconhecimento. Morfotipos de fitólitos são micro corpúsculos de sílica biogênica extraídos de plantas ainda vivas ou de seus restos preservados no solo, e podem ser classificadas em relação às suas características morfológicas. A definição dos métodos de pré-processamento e segmentação das imagens foi realizado a partir de testes utilizando métodos já implementados em programas gratuitos, e para a extração de características é apresentado um conjunto de algoritmos para tratar imagens de *globulares echinate*, morfotipo proveniente da família *Arecaceae*. A partir das ferramentas e algoritmos propostos, foi possível realizar a extração de características das imagens de morfotipos de fitólito, gerando uma base de dados para reconhecimento de padrões. Os métodos de classificação testados não se mostraram tão eficazes na identificação das espécies, devido ao tamanho e problemas de geração da base de dados. Por fim, uma discussão sobre os testes experimentais é apresentada, além dos resultados e conclusões da pesquisa.

Palavras-chave: Processamento de Imagens, Extração de Caraterísticas, Morfotipo de Fitólitos, Globular Echinate.

Capítulo 1

Introdução

Estudos arqueológicos e sua evolução contribuíram de forma significativa no entendimento da sociedade antiga e seus feitos, bem como a evolução tecnológica e artística de cada geração. O desenvolvimento dos estudos fitolíticos das análises de conjuntos e taxonomia de fitólitos contribuiu de forma relacionada permitindo a detecção de alterações climáticas, estrutura de flora e fauna por tempo, bem como indícios das ações humanas na natureza desde a pré-história.

Pesquisadores como biólogos, arqueólogos, geomorfologistas, pedologistas, geógrafos, dentre outros, desenvolveram técnicas de análise de *proxies* ambientais (palinomofos, microrrestos, etc), que focaram a análise de pólen na busca de informações históricas. Porém, esta não pode se embasar e confiar em dados de apenas uma origem metodológica, o que gera a necessidade de obter dados a partir de diferentes análises, na tentativa de reduzir erros e fraquezas de cada método [1].

O estudo de fitólitos permite obter informações relacionadas a quais plantas tem sido utilizadas pelo homem e com qual finalidade, bem como outros dados florestais. Esta análise oferece várias vantagens pelo fato de os fitólitos serem conservados em solos de diferentes níveis de pH, bem como a análise de suas formas, que possibilita rastrear várias informações em nível de família e subfamília de plantas. Produções em grande número destas substâncias também permitem a facilitada análise morfológica e estatística quanto à sua incidência, contribuindo com dados para estudos arqueológicos, a exemplo de [2] e ambientais, como em [3].

Pesquisas como a de [4] apresentam reconstruções histórico ambientais de locais importantes e específicos a partir da análise morfológica de fitólitos, na tentativa de identificar as plantas e suas incidências em solos de diferentes épocas. Além disto, os estudos de [5] mostram que a interpretação dos morfotipos de fitólitos se dá pela análise visual de suas formas e caracterísi-

cas, que são catalogadas referentes à planta da qual foi realizada a extração.

Com a constante e crescente evolução tecnológica é possível definir, desenvolver e prover métodos de reconhecimento de padrões a partir de imagens binárias, que são transformadas a partir do uso de técnicas de processamento de imagens, com a finalidade de auxiliar um especialista em morfologia de fitólitos na tomada de decisão. De acordo com estes conhecimentos, propôs-se neste trabalho a descrição metodológica e implementação de algoritmos para a extração de características de imagens de morfotipos de fitólitos.

A classificação de padrões de imagens digitais a partir da análise de formas visa a extração de características morfológicas das imagens binárias apresentadas, relacionando-as em conjunto, restringindo assim imagens com características próximas a este conjunto [6]. O processo de análise de imagens envolve, segundo [7], descobrimento, identificação e entendimento de padrões, tendo como uma de suas principais metas permitir auxílio computacional aos profissionais da área em foco.

Dentre as abordagens de extração de características, podem ser utilizadas a sintática ou a estatística. Na abordagem sintática, são utilizadas primitivas conhecidas presentes na imagem, onde um algoritmo de junção dessas primitivas deve ser capaz de alcançar a imagem analisada. Já na abordagem estatística são utilizadas medidas como características ou atributos, a fim de permitir propriedades de imagens próximas terem resultados parecidos e vice-versa [6].

O processo de classificação das características já extraídas pode ser realizado de forma supervisionada ou não-supervisionada. Quando no processo de classificação as características das imagens são previamente definidas, o mesmo é chamado de classificação supervisionada. Para a caracterização de cada classe, é necessário uma fase de treinamento a partir de dados previamente identificados, denominado conjunto de treinamento, cada dado possuindo dois componentes, medidas e descrição de um vetor de propriedades e a classe a qual este pertence.

Quando não é conhecido um padrão de classes, caracteriza-se uma classificação não-supervisionada, que segundo [6] se denomina pelo fato de o conjunto de dados não possuir classes definidas. Desta forma, esta abordagem deve rotular os vetores de propriedades, os quais não possuem um significado previamente conhecido.

Entendendo a análise de formas e seu objetivo em definir padrões gerais das morfologias dos fitólitos das espécies da família *Arecaceae*, é indicada a classificação utilizando métodos

de classificação supervisionados, visto que o objetivo da extração de características é gerar um vetor de propriedades para classes conhecidas, que são as espécies.

Com isto, a proposta metodológica de análise de fitólitos apresentada por [2] mostra a possibilidade de reconhecer um morfotipo de fitólito específico da família *Arecaceae*, chamado *globular echinate*, utilizando métricas e análises a partir de métodos de processamento de imagens sob suas formas.

1.1 Objetivos

Propor um modelo de implementação para a proposta de [2], definindo a construção de métodos que se utilizasse de técnicas de processamento de imagens para a extração de características de imagens do morfotipo de fitólito *globular echinate*, encontrado na família *Arecaceae*, obtidas por meio de microscópio petrográfico.

1.2 Organização do Documento

Esta monografia está estruturada da seguinte forma:

O Capítulo 2 apresenta os estudos bibliográficos acerca de Morfotipos de Fitólitos.

O Capítulo 3 aborda os conceitos teóricos de processamento de imagem, oferecendo uma visão macro de seus processos.

O Capítulo 4 define as metodologias e métodos escolhidos e desenvolvidos.

Por fim, os Capítulos 5 e 6 trazem os resultados e discussões e as considerações finais do trabalho, respectivamente.

Capítulo 2

Morfotipos de Fitólito

Segundo os estudos de [8], inúmeros tipos de plantas conhecidas acumulam a chamada sílica sólida ou ácido silícico amorfo ($SiO2$) dentro de células de folhas, sementes, raízes, entre outras partes. Esta substância assume posições e formas diferenciadas, seguindo padrões taxonômicos, sendo denominada de fitólito. Os trabalhos de [9] e [10] afirmam que estes fitólitos, traduzidos literalmente como "planta-pedra", podem se formar em regiões internas ou externas à uma célula, e com a sua solidificação, acaba tomando a forma da célula ou região onde se formou; desta forma, é possível encontrar morfotipos de fitólitos diferenciados. Para [1], os fitólitos, que são produzidos durante o ciclo vegetativo das plantas, são incorporados ao solo e/ou sedimentos após a morte da planta, podendo ali perdurar por longos períodos de tempo.

Como apontado por [1], estudos arqueológicos tem grande atenção focada no homem e no seu ambiente na pré-história, e, para tanto, foram propostas e desenvolvidas técnicas analíticas que auxiliam na análise de elementos, ambientes e sua idade, como a análise de pólen e sementes, porém com alguns erros de precisão do uso destas técnicas.

Um entendimento mais aprofundado sobre a origem das pesquisas na área e uma melhor definição sobre fitólito é apresentada por [10], onde se afirma que os estudos tiveram início em 1896, pelo pedólogo russo Ruprecht acerca da gramínea *Stippa pennata*, seguido de vários outros estudos também citados, que definiram o termo fitólito como sendo corpos silicosos do solo, originado pela decomposição de plantas.

De acordo com [9], quando uma planta morre, seja queimada, enterrada ou ingerida, os fitólitos persistem, e são liberados no meio ambiente, mantendo a integridade de sua forma, se tornando então um microrresto da planta de origem. Estes fósseis podem ser extraídos de várias fontes, como a própria planta, solos e sedimentos, bem como de artefatos históricos criados e

manuseados pelo homem, o que permite o uso destes corpúsculos em estudos etnobotânicos e paleoeconomicos.

Segundo [11], os fitólitos possuem características morfológicas, que permitem diferenciação entre famílias, subfamílias e, por vezes, gênero, possibilitando a identificação de plantas. Neste contexto, morfotipos de fitólitos podem ser usados para fins históricos e culturais (etnobotânica) e de compreensão de condições ambientais, permitindo reconstruções vegetais ao longo do tempo geológico, bem como inferir dados sobre clima em escala local ou regional no passado.

De acordo com [10], reconstruir a vegetação de uma área a partir dos morfotipos de fitólitos se baseia em uma fase de extração local da substância encontrada no solo. A seguir, se faz a retirada e catalogação destas partículas/corpúsculos a partir de plantas existentes na atualidade (coleções de referência), e por fim, utiliza-se estes dados para comparação dos tipos de corpúsculos obtidos desses dois materiais, solo e planta, a partir de suas características morfológicas, determinando, aproximadamente, a qual família de planta pertence o morfotipo proveniente do solo.

Os resultados do trabalho de [1], que busca encontrar diferenças entre os morfotipos de fitólito de forma que seja possível a identificação de sua origem, mostram que há muita multiplicidade entre os morfotipos, havendo a aparição das mesmas formas em diferentes plantas, além de uma mesma planta conter inúmeros fitólitos diferenciados. Ele afirma que a identificação da planta não é possível ao mesmo nível que a análise de pólen, e que não há relações entre diferenças morfológicas e nível taxonômico. Porém, este autor afirma que seu trabalho não garante a ineficácia do uso de fitólitos para a finalidade de identificação da planta, mas sim que alguns morfotipos não acrescentam qualquer valor na diferenciação de níveis taxonômicos. Já o trabalho de [2], que é mais recente, mostra resultados mais satisfatórios em relação a identificação das espécies da família *Arecaceae*, que será detalhado na sequência.

O trabalho de [5], apresenta que o processo de extração dos morfotipos de fitólitos do solo se dá inicialmente pela separação destes dos outros componentes minerais e orgânicos presentes no solo a partir da eliminação da matéria orgânica do solo e da solubilização e remoção dos óxidos e hidróxidos de ferro e alumínio do solo. Em seguida é necessário um fracionamento físico por meio de flutuação dos fitólitos, usando um líquido pesado (densidade 2.35). Desta

extração por flutuação, se obtém os fitólitos, diatomáceas e outros corpos silicosos, também são recuperadoos. Este material é preparado em lâminas imersas em óleo para observações em 3D. Após estes passos, é efetuada a análise e interpretação morfológica dos fitólitos para obtenção de informações deste conjunto de morfotipos do solo. A interpretação do conjunto de fitólitos permite inferir as condições de umidade (índice fitolítico - Iph), de temperatura (índice climático - IC) e grau de cobertura arbórea (IDP) que predominavam ao longo da pedogênese do solo estudado.

Os estudos de [2] mostram que, a partir de morfotipos de fitólitos produzidos por palmeiras (família *Arecaceae*), *spheroid* (*globular*) *echinate*, é possível definir um padrão de medidas que apresente/demonstre uma variação pequena entre os morfotipos produzidos pelas espécies desta família. Os autores relatam outros estudos, mostrando ser possível definir métodos computacionais que sejam capazes de extrair características genéricas de medida para o fitólito especificado, facilitando os processos de identificação. Os mesmos afirmam que a incidência do fitólito *globular echinate* não se restringe apenas à família *Arecaceae*, podendo estar presente em outras famílias, como por exemplo em gramíneas.

Em [2], é colocado que a análise morfométrica de fitólitos é um efetivo diferenciador entre fitólitos produzidos por taxas próximas. Em seu trabalho, cerca de 30 a 50 fotos de morfotipos de diferentes origens foram utilizadas para a extração manual de medidas, que se seguiu da projeção de uma elipse externa, incluindo as protuberâncias do morfotipo, e outra interna sem a inclusão das protuberâncias. Com estes dois objetos, foram calculados os parâmetros da diferenças entre as áreas das elipses, indicando a altura das protuberâncias, e as medidas dos eixos maior e menor da elipse interna, cuja razão indica o nível de esfericidade do morfotipo. A Figura 2.1 mostra a imagem modelo de extração destas características apresentado pelos autores.

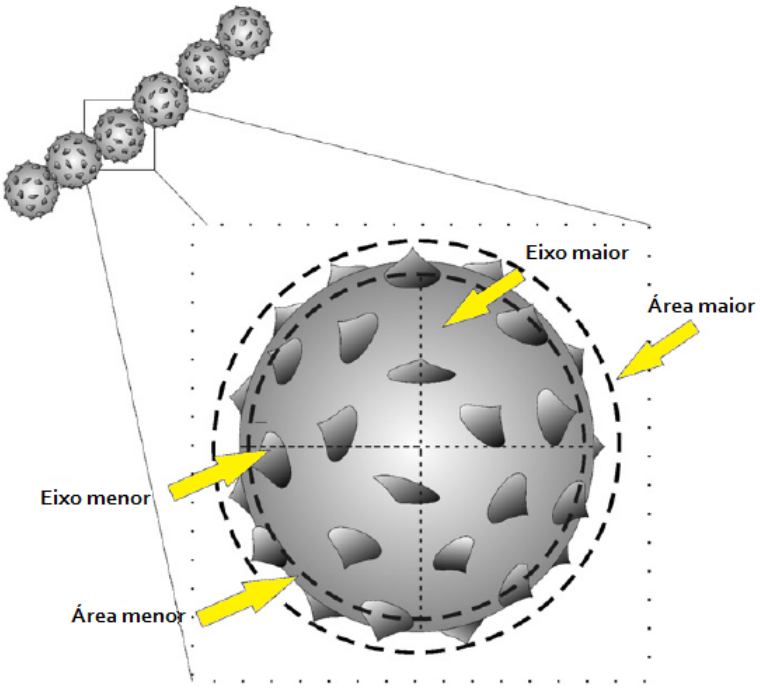

Figura 2.1: Fotomicrografia da morfologia de *globular echinate* da espécie *Phoenix Reclinata*, em 400 x, e o desenho de um fitólito de palmeira mostrando os parâmetros selecionados para a retirada de medidas morfométricas

Adaptado de [2]

A partir dos conceitos apresentados, entende-se que medidas podem ser definidas a ponto de diferenciar os fitólitos *globular echinate* das espécies da família *Arecaceae*, de forma que o trabalho de reconhecimento possa ser otimizado para os especialistas e estudantes da área.

Inúmeras técnicas de transformações de imagens digitais, bem como a extração de características e sua análise podem ser definidas para alcançar a finalidade de prover processos de classificação de padrões confiáveis, podendo alcançar resultados diferenciados. Para definir uma ferramenta satisfatória, é necessário entender do problema e das imagens, e então, a partir das teorias, definir passos para a criação de uma ferramenta de classificação apresentando resultados confiáveis, que seja realmente utilizada por pesquisadores especialistas.

Este capítulo apresentou brevemente conceitos sobre fitólitos bem como trabalhos que buscam catalogar e identificar seus morfotipos a partir de padrões, sempre definidos e estudados de forma manual, que buscam facilitar os trabalhos de interpretação dos conjuntos de fitólitos extraídos do solo, dos quais não se sabe previamente de qual planta provem tais morfotipos.

Desta forma, é visível a importância do trabalho e a contribuição da computação nos pro-

cessos de análise de morfotipos de fitólito, principalmente a partir da metodologia proposta por [2]. O próximo Capítulo apresenta os conceitos e modelos computacionais a serem utilizados para o desenvolvimento do trabalho, e estes resultados otimizarão as análises e reconstruções culturais e ambientais, pautados no princípio do uniformismo.

Capítulo 3

Processamento de Imagens

A representação e manipulação de uma imagem digital se dá por meio da interpretação de dados numéricos na forma de cores e pontos, que necessitam de um componente capaz de converter uma cena real numa imagem discretizada matematicamente, que é uma imagem digital.

Uma imagem digital do tipo discreta possui suas cores de forma disposta num plano de pontos (x, y), cujo agrupamento destes pontos pode ser dado por uma matriz nxm representando a imagem, onde cada posição possui uma contribuição RGB (*Red*, *Green*, *Blue*) para a imagem naquele ponto. Desta forma, se torna facilitada a interpretação matemática, e ao mesmo tempo limita a qualidade da imagem final, muitas vezes não percebida pela visão humana [6].

O processamento de uma imagem se dá então pela manipulação das informações de cor de cada ponto da imagem, interpretando-os ou alterando seu valor a depender do objetivo a ser alcançado. Aproveitando o fato de que a informação de cor de um ponto influencia a região em que se encontra, muitos métodos utilizam as informações dos pontos vizinhos a este, facilitando tanto a interpretação quanto a alteração dos valores, mantendo-os de acordo com o padrão da imagem. Para facilitar o processo de manipulação dos pontos da imagem, são utilizadas máscaras para alterar cada ponto tomando seus vizinhos durante o processo, que podem ser representadas por uma matriz quadrada (mxm).

Processos de modificação dos pontos de cor da imagem são muito custosos computacionalmente, com operações de soma, multiplicação e divisão sucessivas ponto a ponto. Uma máscara matricial pode ser preparada com várias operações conjuntas a serem atribuídas a cada ponto da imagem, diminuindo em parte o custo computacional, utilizando-a em conjunto com o conceito de convolução. Entendendo uma imagem e seus pontos como uma matriz nxm de dimensões

desconhecidas, pode-se aplicar um processamento por uma máscara matricial quadrada de tamanho conhecido. Para cada ponto da matriz da imagem, toma-se uma sub-matriz desta com as dimensões da máscara, mantendo o ponto em foco no centro desta sub-matriz, e assim operando cada posição da sub-matriz com a correspondente posição da máscara, atribuindo (somando) todos os valores resultantes no ponto final, que é o central em relação a sub-matriz quadrada da imagem. Tomando este conceito, métodos de processamento de imagem definem matrizes de máscara para convolução na imagem gerando o resultado esperado [6][1].

3.1 Sistema de Visão Computacional

Um programa ou algoritmo para processamento, extração de dados e reconhecimento de imagens digitais é denominado como sistema de visão computacional, que é divido em processos ou etapas, que devem ser estudados e testados para garantir a eficácia de cada etapa no reconhecimento das imagens a serem manipuladas [7].

De acorco com [6], o processamento de imagens é definido em cinco etapas, conforme a Figura 3.1.

A aquisição de uma imagem se dá pela sua captura a partir de um dispositivo que seja capaz de convertê-la para uma representação adequada ao processamento.

Ainda segundo [6], imagens digitais representadas por monitores e câmeras se utilizam comumente do padrão de cores RGB, que é baseado em três valores componentes de intensidade de uma cor, o vermelho (R), verde (G) e azul (B), a fim de representar a cor desejada a partir da contribuição de cada valor na sua intensidade. Esta escala é representada por um cubo, onde a origem gera a cor preta, o vértice mais distante gera o branco e o vetor gerado por estes pontos indica a escala de cinza.

Uma imagem adquirida pode apresentar falhas ou ruídos devido às condições em que foi capturada ou concebida, portanto, a etapa seguinte que é a de pré-processamento busca melhorar a qualidade desta imagem a partir de técnicas de ajuste, seja de ruídos, brilho, contraste ou qualquer outra operação que aprimore as suas propriedades, para posterior aplicação.

[1] O processo descrito se dá no domínio espacial, mas também pode ser aplicado no domínio da frequência, a partir da transformada de Fourier, por exemplo.

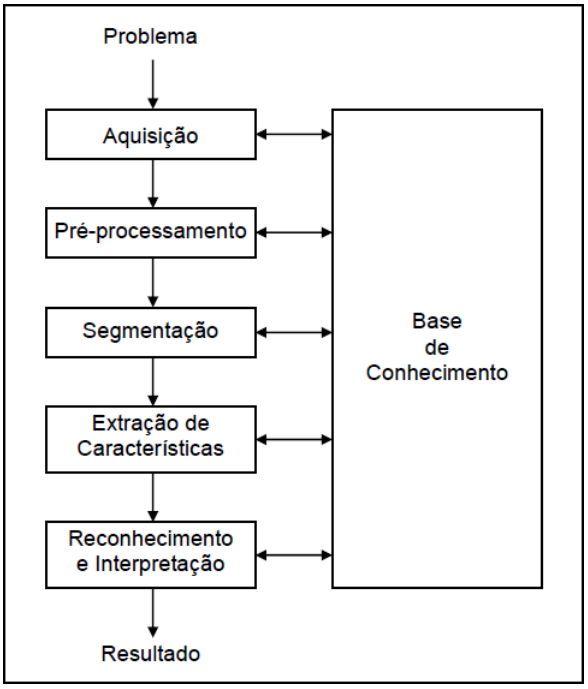

Figura 3.1: Etapas do processamento de imagens
[6]

No início da etapa de pré-processamento, é necessário o uso do processo de quantização. A quantização de uma imagem para escala de cores em tons de cinza, ou seja, diminuir a gama de cores da imagem para 256 tons entre preto e branco, é uma técnica por vezes essencial nas etapas de processamento de imagem. Ao mesmo tempo não se faz necessária a imposição de que a imagem de entrada para o sistema já esteja convertida para tons de cinza, uma vez entendendo que o processo para esta transformação não traz custo relevante ao processamento e permite uma abertura ao usuário em não se preocupar com esta preparação.

Por fim, o pré-processamento busca melhorar a qualidade visual da imagem, não aumentando sua quantidade de informações, mas apenas tornando mais fácil a percepção de detalhes e elementos de uma imagem [7].

Em seguida, vem a etapa de segmentação ou *thresholding* que realiza a separação dos elementos da imagem, mantendo as áreas de interesse a partir de métodos de detecção de bordas ou regiões. Esta etapa é de fundamental importância para que as informações resultantes do sistema de processamento de imagens sejam de fato confiáveis.

O processamento de uma imagem com o objetivo de segmentar certo número de objetos que possam apresentar posições, tamanhos e formas diferentes é uma tarefa difícil e que depende

da correta extração. A presença de ruídos em uma imagem pode levar os métodos de segmentação a distorcer as formas dos objetos encontrados, permitindo que a forma encontrada seja identificada como próxima dentre outras completamente diferentes.

Como afirmam [6], esta etapa de separação de objetos trabalha em geral nos níveis de cinza de uma imagem, utilizando técnicas de reconhecimento de descontinuidades ou similaridades da imagem. Técnicas de descontinuidade buscam separar a imagem com base nas mais intensas mudanças dos níveis de cinza, caracterizando a presença de pontos isolados, linhas ou bordas nesta imagem. Já as técnicas de similaridade buscam conjuntos de pontos que apresentem valores próximos (similares) a partir de um determinado conjunto de características. Em ambos os casos, é necessário o mapeamento de características esperados da imagem, que está diretamente relacionado ao domínio da aplicação.

A representação da imagem adquirida pela etapa de segmentação é utilizada para armazenar e manipular as regiões de interesse mantidas, enquanto a descrição visa a extração de características ou propriedades de forma que seja possível diferenciar estas áreas de interesse. Este último processo pode ser descrito por atributos, medidas ou vetores de características [7].

A fase de segmentação pode ser também denominada de limiarização, que é divida em dois grupos, dos métodos locais e globais. Na limiarização local a imagem total é dividida em setores de tamanhos iguais, aplicando a limiarização de forma distinta em cada setor. Já na abordagem global, a imagem é tomada como um todo e a limiarização é aplicada unicamente.

Segundo [6], a limiarização ou *thresholding* objetiva-se na classificação dos pixels de uma imagem de acordo com seus limiares. Este método consiste no uso de uma constante limiar T e uma função $f(x, y)$ tal que se num dado ponto (x, y), a expressão $f(x, y) > T$ denomina um ponto do objeto, e analogamente, a expressão $f(x, y) < T$ denomina um ponto de fundo. Desta forma a imagem é binarizada, ou seja, são utilizadas apenas duas cores, uma para definição de objeto e outra para fundo.

Tanto na limiarização local como na global é utilizado o histograma da imagem ou subimagem. Um histograma corresponde a um gráfico de distribuição de ocorrência de uma determinada cor sobre a imagem, que geralmente são em tons de cinza. Sua amostragem traz informações acerca de seu brilho e contraste.

Na limiarização global, é selecionado como constante limiar o valor que corresponda ao

nível de cinza com objetivo de melhor dividir os tons de cinza em dois grupos no histograma da imagem, a serem determinados como pontos de objeto e pontos de fundo. Segundo [6], há desvantagens em se utilizar o limiar global, pois nem sempre as intensidades dos objetos e do fundo são bem distintas, como a ocorrência de baixo contraste, ocasionando na perda de características da imagem. Assim, a escolha de uma constante limiar não é uma tarefa trivial, tendo disponíveis várias técnicas que alcançam bons resultados, dependendo do estado da imagem. Dois métodos de limiarização serão discutidos no próximo capítulo, o comum, que utiliza uma constante limiar fixa, e o método proposto por [12], que utiliza os dados do histograma para definir uma constante limiar ideal dinamicamente.

Ainda para a fase de segmentação, é necessário isolar apenas as bordas do objeto, para que seja facilitada a interpretação ou extração de características do objeto na fase seguinte. Em [7], é definido que o uso da detecção de bordas é a melhor abordagem dentre a detecção de descontinuidades, pois é a situação mais decorrente. Os tipos de detecção de descontinuidades são colocadas como detecção de linha, ponto e borda.

Tanto [6] quanto [7] explicam a fase de detecção de bordas com os mesmos princípios, afirmando que uma borda é o limite ou fronteira entre duas regiões que apresentem diferenças relativamente distintas. Um detector trabalha com o uso de um operador local diferencial, ou seja, realiza o uso de primeira e segunda derivadas para definir as descontinuidades com valores positivos e negativos para ambos os operadores, e valores nulos quando os tons considerados são constantes. Estas operações são realizadas em subespaços de uma imagem, detectando as bordas localmente.

Por fim, o reconhecimento ou classificação de padrões, atribui um identificador à região descrita, e a partir disso o processo de interpretação realiza o trabalho de separar estes identificadores de forma que seja possível aproximar as imagens que possuem regiões de interesse muito parecidas.

A classificação de padrões de imagens digitais a partir da análise de formas, que compreende à última etapa visa a extração de características morfológicas de amostras relacionadas em conjunto, restringindo imagens com características próximas a este conjunto [6]. O processo de análise de imagens envolve descobrimento, identificação e entendimento de padrões, tendo como uma de suas principais metas prover ao computador a capacidade de aproximar-se da assi-

milação humana, ou seja, a tentativa de realizar tarefas analíticas relacionadas a imagens como faria um especialista [7].

Todas estas etapas são codificadas na forma de uma base de conhecimento, que pode variar em tamanho e complexidade dependendo da aplicação e do contexto envolvido.

O trabalho de [7] define o processamento de imagens de forma reduzida, focando nas etapas de pré-processamento, segmentação unido à extração de características e interpretação, denominando as três fases de processamento de baixo nível, nível intermediário e alto nível. No processamento de baixo nível, são utilizadas técnicas de transformações de imagens, também antes como pré-processamento, preparando a imagem de forma que seja possível extrair informações. O processamento de nível intermediário se dá pela extração das características ou informações da imagem após este pré-processamento, utilizando técnicas de segmentação e descrição. Por fim, no processamento de alto nível são interpretados os dados extraídos, a fim de classificá-los a partir de métodos como utilização padrões pré-definidos ou comparação com extrações anteriormente realizadas.

O estudo sobre processamento de imagens e definição de um sistema de visão computacional permite a compreensão da complexidade da aplicação, desde o recebimento da imagem até a concepção da interpretação desta para a geração de conhecimento. No contexto deste trabalho, a base de conhecimento apresentada a ser implementada prevê o recebimento de uma imagem de entrada de um morfotipo de fitólito isolado, juntamente com uma barra de escala para ajuste de dados. Estes processos de implementação podem ainda apresentar necessidades de preparação para a imagem de entrada.

Com este objeto, o algoritmo deve ser capaz de realizar as transformações necessárias para a extração das bordas do fitólito bem como da barra de escala, e, para o primeiro, realizar a projeção interna e externa de elipses para a extração de medidas conforme propõe o trabalho de [2], e feito isso, utilizar o tamanho da barra para ajuste e normalização de escala. Desta forma, os dados de saída esperados são as medidas dos eixos maior e menor da elipse interna e a distância entre áreas da elipse interna e externa, para então ser utilizada pela fase de reconhecimento e interpretação nos moldes do sistema de visão computacional. Uma apresentação mais detalhada sobre os dados a serem manipulados pela base de conhecimento será feita no Capítulo 4.

Capítulo 4

Materiais e Métodos

Este capítulo tem por objetivo apresentar os objetos de teste e os métodos escolhidos e desenvolvidos para uso neste trabalho. No Capítulo 3 foram apresentados os passos para a construção de um sistema de processamento de imagens, onde uma imagem deve passar por determinados tratamentos até que se construa um conjunto de informações ou vetor de propriedades, para que assim possam ser diferenciadas e identificadas. Desta forma, serão definidos os métodos específicos para tratar o conjunto de imagens dos *globulares echinate*.

As imagens utilizadas nos parâmetros de testes durante o trabalho fazem parte do banco de dados gerado pelo grupo de pesquisa "Gênese e Evolução de Superfícies Geomórficas e Formações Superficiais", formado no Curso de Geografia na Unioeste nos campus de Marechal Cândido Rondon e Francisco Beltrão. Estas imagens foram adquiridas em microscópio ótico petrográfico, equipados com uma câmera capaz de converter imagens das laminas para um modelo manipulável computacional contendo morfotipos de fitólitos, que são montados em lâminas com óleo de imersão, amostrados conforme [5].

No processo de construção de um sistema para processamento de imagens, pode-se esperar que a imagem passe por um processo de preparação, tanto na alteração de cores e brilho, quanto nas suas dimensões. Entendendo a necessidade de facilitar os processos realizados pelo usuário, não foram especificadas quaisquer preparações, porém, há a percepção de que estas são adquiridas sem o controle de aproximação, ou seja, a escala dos objetos tende a variar, o que dificulta a interpretação dos dados extraídos. A imagem esperada para o sistema deverá conter somente o objeto a ser identificado juntamente com uma barra de escala na medida de 10 micrômetros, evitando o uso de algoritmos para ajuste de escala, tornando esta conversão de medidas mais confiável.

A Figura 4.1 mostra a imagem adquirida de um microscópio contendo inúmeros fitólitos da espécie *Bactris Caryotaefolia*, e a Figura 4.2 apresenta uma imagem de *globular echinate*, apresentando os padrões esperados.

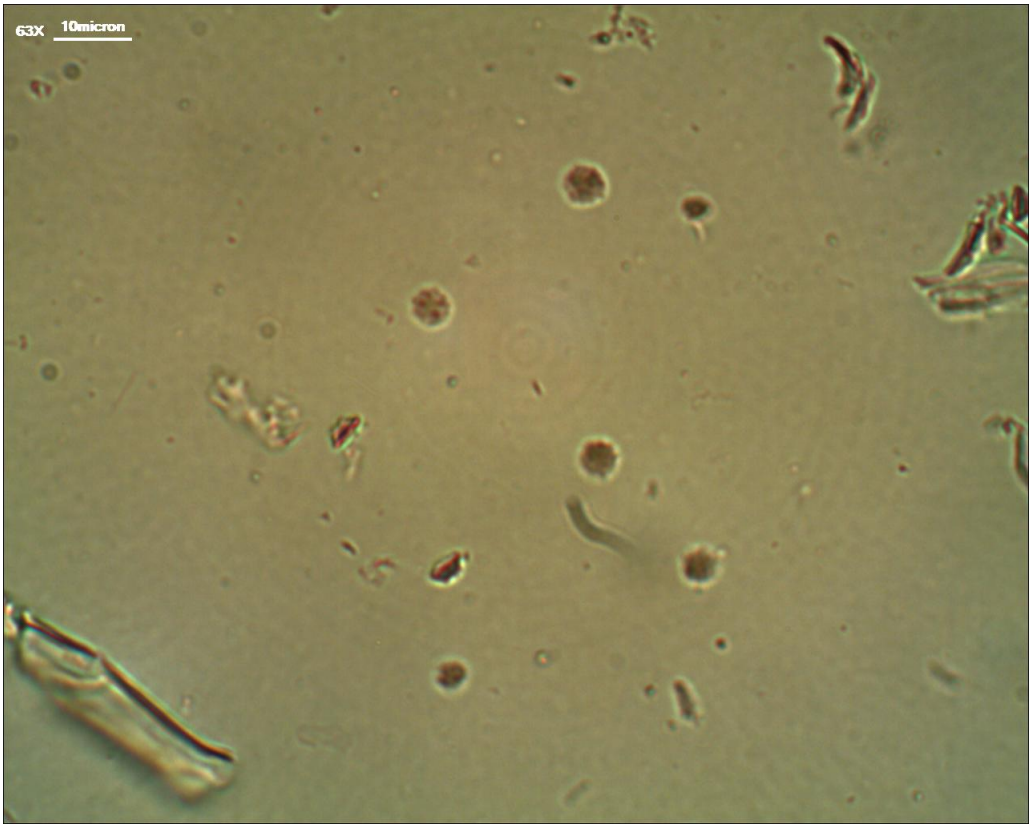

Figura 4.1: Imagem adquirida de microscópios contendo vários fitólitos da espécie *Bactris Caryotaefolia*

Figura 4.2: Imagem de um *globular echinate* da espécie *Bactris Caryotaefolia*, isolada manualmente com medida de escala

Para a realização de testes e preparação das imagens antes da escolha das técnicas para implementação, foram utilizados os programas ImageAnalyzer, PDITools, IPTool e ImageJ. Para a finalização dos testes e implementação foi utilizada a biblioteca OpenCV definida por

[13], que será apresentada na seção seguinte. A utilização dos programas citados contribuiu para a compreensão do comportamento dos métodos de processamento de imagens analisados em relação às imagens de morfotipos de fitólitos antes da escolha das técnicas que realmente trariam resultados satisfatórios.

4.1 Biblioteca OpenCV

OpenCV (*Open Source Computer Vision*), apresentada por [13], é uma biblioteca gratuita sob a licença BSD (*Berkeley Software Distribution*), oferecendo funções e métodos para sistemas de visão computacionais em tempo real, atualmente disponível para desenvolvimento nas linguagens de programação C, C++ e Python. O website oficial (http://opencv.willowgarage.com/wiki/) oferece versões para diferentes sistemas operacionais, facilitando ainda a portabilidade do código desenvolvido, e uma documentação completa da biblioteca, que demonstra a utilização dos parâmetros e como é a funcionalidade e origem da implementação de cada método.

O uso desta biblioteca facilita não apenas na utilização de vários métodos sem a necessidade de implementação, que poderia levar a problemas, mas também mantém um padrão de estruturas de dados e de bom desempenho dos algoritmos que tratam a imagem em memória.

A biblioteca foi utilizada na versão 2.4.0, juntamente com a linguagem de programação C++ no ambiente Ubuntu Linux versão 12.04 LTS. Foram utilizadas as estruturas de dados oferecidas pela biblioteca e métodos disponíveis para a fase de pré-processamento e segmentação. Já para a fase de extração de características, por se tratar de um caso específico, foi desenvolvido um conjunto de métodos para efetuar as operações, descrito na sequência deste documento.

4.2 Pré-processamento e Segmentação

Inicialmente, foram realizados experimentos com várias imagens do morfotipo de fitólito *globular echinate*, submetendo-as a diferentes métodos de processamento de imagem, com o objetivo de realçar suas características de forma a otimizar a detecção das formas. Os métodos utilizados para os testes de preparação e extração de bordas da imagem, ou seja, com apenas pixels na cor branca representando as bordas do objeto, com o resto da imagem em cor preta correspondendo ao fundo da imagem, foram:

- Quantização para escala em tons de cinza;
- Limiarização estática;
- Limiarização utilizando o método binomial de Otsu;
- Detecção de bordas utilizando o método de Canny;
- Detecção de círculos pela transformada de Hough;
- Detecção de bordas pelo método de Roberts;
- Detecção de bordas pelo método de Prewitt;
- Detecção de bordas pelo método de Sobel;
- Detecção de bordas pelo método Isotrópico;
- Detecção de bordas pelo método de Kirsch;
- Detecção de bordas pelo método de Laplace;
- Detecção de bordas pelo método de Gabor;
- Detecção de bordas pelo algoritmo Método de análise por acompanhamento de borda.

Como explicado no Capítulo 3, é interessante utilizar primeiramente a conversão da imagem para tons de cinza, para evidenciar os elementos da imagem para a limiarização e detecção de bordas, compondo assim a fase de pré-processamento. A Figura 4.3 apresenta um *globular echinate* com a aplicação da conversão em tons de cinza.

Figura 4.3: Aplicação de conversão em escala de tons de cinza para a imagem de um *globular echinate* da espécie *Poliandrococo Caudensis*

Seguindo à fase de segmentação, foram utilizados duas abordagens, a limiarização por uma constante limiar fixa e pela utilização do método proposto por [12], que apresenta uma estratégia interessante, pois como o limiar é selecionado dinamicamente, pode-se dizer que é uma forma adaptativa de se isolar os objetos de uma imagem. Porém, como as imagens de *globular echinate* para uso são bastante específicas, foi percebido que em todos os casos o nível de ruídos mantidos eram muito altos, o que dificultava bastante para as fases seguintes. A Figura 4.4 (a) apresenta a imagem de um *globular echinate* após a aplicação da limiarização pelo método de Otsu. O uso da limiarização por uma constante foi capaz de retirar melhor os ruídos da imagem, utilizando um fator $T = 100$, cujos resultados são apresentados na Figura 4.4 (b).

Além da limiarização para separar o fitólito, é necessário realizar a separação da barra de escala da imagem. Como a barra possui a cor branca, a utilização do limiar fixo $T = 210$, que é um valor bastante alto, garantiu a separação da barra de escala por completo, como mostra a Figura 4.4 (c).

Figura 4.4: Aplicação da limiarização pelo método: (a) binomial de Otsu; (b) limiar fixo; (c) limiar fixo para a extração da barra de escala, sob a Figura 4.3

Para a detecção de bordas do objeto, os métodos citados foram utilizados na fase de testes com os programas auxiliares, também citados. Como resultado, foi verificado que os métodos de Roberts, Prewitt, Sobel e o Isotrópico apresentaram resultados equivalentes, que não foram suficientemente satisfatórios, pois foi possível perceber falhas na continuidade das bordas do objeto, mostradas na Figura 4.5 (a), que poderiam trazer problemas para a extração do vetor de propriedades. Alem destes, os testes utilizando o método de Kirsch mantiveram muitos ruídos de borda, impossibilitando a identificação do objeto de análise, como mostra a Figura 4.5 (b).

Os métodos de detecção de bordas que obtiveram os melhores resultados foram o de Laplace, mostrado na Figura 4.5 (c), Canny [14], apresentado na Figura 4.5 (d), e o de Método de análise por acompanhamento de borda proposto por [15], mostrado na Figura 4.5 (e). Os

métodos de Laplace e Canny apresentaram resultados bastante parecidos, com boa definições de borda e uma quantidade baixa de ruídos em relação aos outros métodos. Porém, como apresentado, é visível a diferença para o método Suzuki, que se utilizou da limiarização estática em diferença aos outros, sendo capaz de reduzir por completo os ruídos, mas que não garante este melhor caso para todas as imagens.

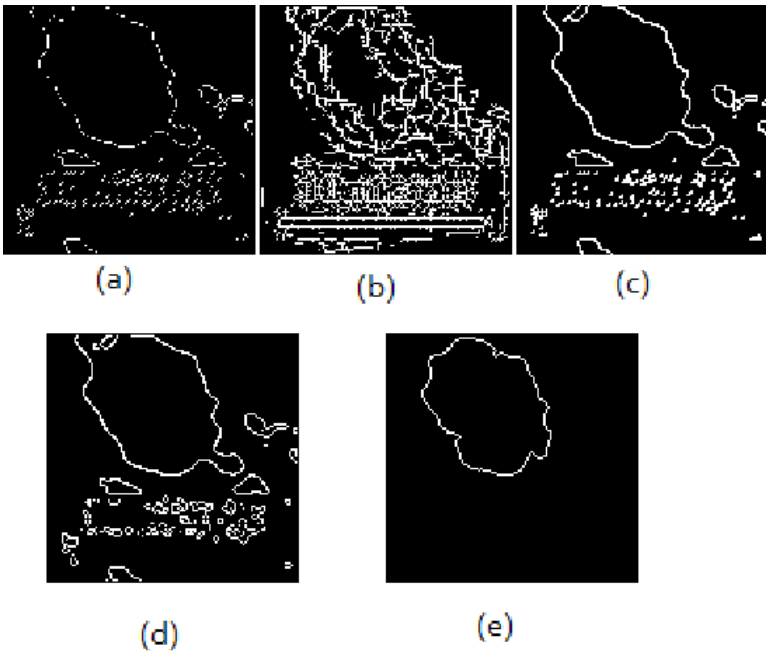

Figura 4.5: Aplicação da detecção de bordas pelos métodos de: (a) Roberts, Prewitt, Sobel e o Isotrópico; (b) método de Kirsch; (c) método de Laplace; (d) método de Canny, sob a Figura 4.4 (a), e (e) método de Suzuki sob a Figura 4.4 (b)

A partir dos métodos analisados e testados, foram selecionados a limiarização por uma constante limiar fixa, tanto para a detecção do objeto, quanto para a medição da barra para ajuste de escala, e em sequência, manteve-se o uso do método de detecção de bordas proposto por [15]. Estes métodos concluem a fase de segmentação da imagem.

O método proposto por Suzuki utiliza a técnica de detecção de bordas por caminhamento, do qual deriva uma sequência de coordenadas de pontos de borda conectados de 1-pixel (pontos de objeto), e outra de 0-pixel (pontos de fundo). Este conjunto de pontos de borda é encontrado a partir de um caminhamento no objeto da imagem, buscando manter uma conexão entre os pixels de borda até que esta conexão se feche, formando um objeto. Por fim, este método é capaz de encontrar e isolar vários objetos contidos na imagem, incluindo possíveis ruídos,

porém, pelo fato de separá-los é possível adquirir o conjunto de pontos de apenas um objeto e, dada uma imagem que apresenta apenas o objeto de interesse e ruídos menores, sabe-se que o maior conjunto pertencerá a este objeto.

Neste ponto, o sistema de visão computacional possui como resultado do método de Suzuki, implementado na biblioteca OpenCV apresentado por [13], uma lista ou vetor contendo os pontos de borda, tanto do objeto quando da barra de escala. Como já afirmado, a fase de extração de características é de cunho bastante específica, não havendo técnicas e métodos apropriados que realizem as medições necessárias apresentadas por [2]. A sessão seguinte apresenta os métodos desenvolvidos para esta fase.

4.3 Extração de Características

Para que seja possível alcançar os cálculos e medidas apresentados por [2], primeiramente se fez necessária a projeção das elipses no morfotipo adquirido dos processos anteriormente apresentados e, entendendo o maior interesse nas medidas da elipse interna, esta deveria ser a primeira a ser alcançada, cujos requisitos são tais que a elipse fique totalmente interna a todos os pontos do fitólito e que seja a maior possível. A Figura 4.6 mostra a projeção alcançada pelos métodos implementados.

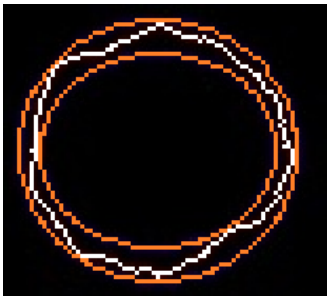

Figura 4.6: Aplicação dos algoritmos de extração de características numa imagem da espécie *Bactris Bahienses*

Durante a implementação do Algoritmo 1, que sofreu várias alterações pelo uso de diferentes metodologias não só de conceitos provenientes de processamento de imagens, mas também da Matemática, não foi encontrado um método que garantisse a melhor projeção possível de uma elipse, ou seja, todos os métodos abordados geravam um erro conforme a imagem tivesse uma esfericidade menor, fato percebido visualmente.

O Algoritmo 1 tem como entrada dois parâmetros, uma lista dos pontos do fitólito e o maior retângulo que abrange a estes pontos, ambos adquiridos a partir dos métodos disponíveis na biblioteca OpenCV [13]. Logo de início é realizada uma rotação dos pontos do fitólito e do retângulo, para que ambos fiquem no sentido horizontal em relação à maior reta do retângulo, de forma a melhorar a compreensão da imagem e melhorar a precisão do algoritmo. Em seguida, é gerada a menor elipse para o ponto central do retângulo como chute inicial pelo algoritmo da menor elipse. Com os dados de largura e altura desta elipse, é realizada uma medição com todos os pontos do fitólito para encontrar o mais próximo desta elipse, e assim definir em qual quadrante houve esta ocorrência, para que o centro seja movimentado no sentido contrário a este, com o objetivo de aumentar a elipse. Para esta nova medição, é feita a chamada do método recursivo apresentado em Algoritmo 2, que dá continuidade à metodologia de movimentação do centro de forma recursiva, até que não seja possível aumentar a área da elipse. As áreas resultantes são comparadas, e a maior delas é apresentada.

O método recursivo apresentado no Algoritmo 2 realiza os mesmos cálculos do Algoritmo 1, com a diferença de que este realiza três chamadas recursivas utilizando alterações na posição do ponto central, que são, respectivamente, a movimentação em x ao quadrante contrário ao que houve o ponto mais próximo da elipse, a movimentação em y da mesma condição, e por fim, a movimentação em x e y (diagonal). O critério de parada da recursividade ocorre quando a área encontrada pelo Algoritmo 3 não possui aumento em relação à área anterior, retornando ao Algoritmo 1 a maior área encontrada, para fins de comparação.

O Algoritmo 3 trabalha com a lista de pontos do fitólito, o ponto central do retângulo e sua largura e altura, que geram uma elipse interna a todos os pontos. Após a translação dos pontos à origem, são realizadas duas diminuições, de largura e de altura respectivamente, onde a largura é diminuída em uma unidade quando a altura foi diminuída até sua metade, que indica que apenas a diminuição de altura não permite a elipse conter todos os pontos. Este processo é repetido até que as medidas da elipse sejam suficientes para conter todos os pontos do fitólito.

O Algoritmo 4 realiza as mesmas operações que o Algoritmo 1, com a diferença de que a elipse é aumentada para encontrar a solução, e o critério de parada é a elipse conter todos os pontos. A partir desses algoritmos, é possível realizar a projeção das duas elipses, e utilizar suas medidas para derivar os valores utilizados na metodologia proposta por [2].

Algoritmo 1 Projeta a Elipse Interna

Entrada: Lista de pontos de borda, $contorno = \{P1, \ldots, Pn\}$.
Retângulo de ajuste inicial aos pontos, $centro\{x, y\}, angulo, tamanho\{altura, largura\}$.
Variáveis:

1: $p1, p2, ponto, c1, c2$: **Ponto**
2: i : **inteiro**
Início
3: $p1 \leftarrow$ ponto da extremidade de uma lateral menor do retângulo
4: $p2 \leftarrow$ ponto da extremidade da outra lateral menor do retângulo
5: Rotaciona a imagem de acordo com o lado maior do retângulo
6: Ângulo de rotação: $tg((p1.y - p2.y)/(p1.x - p2.x))$
7: Chama o Algoritmo da menor elipse projetando a menor elipse para o ponto central do retângulo
8: Calcula a área da elipse encontrada
9: Translada os pontos do $contorno$ para a origem em relação ao centro do retângulo
10: Busca o ponto que está mais próximo da elipse
11: **for** $i = 0 \rightarrow contorno.tamanho$ **do**
12: Calcula a distancia do ponto em $contorno[i]$ em relação à borda da elipse pela equação da elipse: $(contorno[i].x^2)/((largura/2)^2) + (contorno[i].y^2)/((largura/2)^2)$
13: **if** $distancia < menorDistancia$ **then**
14: $menorDistancia \leftarrow distancia$
15: $ponto \leftarrow contorno[i]$
16: **end if**
17: **end for**
18: Translada os pontos do $contorno$ para a sua posição original
19: $c1 \leftarrow centro$
20: $c2 \leftarrow centro$
21: Verifica a qual quadrante pertence o $ponto$ encontrado em $contorno$, e movimenta $c1$ e $c2$ para as direções contrárias
22: $c1 \leftarrow$ retorno da chamada para o Algoritmo recursivo com ponteiro de $area1$
23: $c2 \leftarrow$ retorno da chamada para o Algoritmo recursivo com ponteiro de $area2$
24: **if** $area \leqslant area1$ **and** $area \leqslant area2$ **then**
25: **if** $area1 \geqslant area2$ **then**
26: $area \leftarrow area1$
27: $centro \leftarrow c1$
28: **else**
29: $area \leftarrow area2$
30: $centro \leftarrow c2$
31: **end if**
32: **end if**
33: chamada para o Algoritmo da menor elipse para o ponto de centro encontrado
34: **print** "$Largura$ da elipse final"
35: **print** "$Altura$ da elipse final"
36: **print** "$area$ da elipse final"
Fim.

Algoritmo 2 Chamada Recursiva para o Método da Elipse Interna

Entrada: Lista de pontos de borda, $contorno = \{P1, \ldots, Pn\}$,
Ponto do centro a ser utilizado na iteração, $centro$,
Largura e Altura de referência para diminuição, $largura$, $altura$,
Área calculada na iteração anterior e Área para retorno como ponteiro, $areaAtual$, $area$
Saída: O ponto central localmente selecionado

Variáveis:

$ponto, c1, c2, c3$: **Ponto**
$ar, area1, area2, area3$: **real**
i : **inteiro**

Início

Chamada para o Algoritmo da menor elipse para o ponto de centro de entrada
Calcula a área da elipse em ar
if $ar > areaAtual$ **then**
    Translada os pontos do $contorno$ para a origem em relação ao centro do retângulo
    Busca o ponto que está mais próximo da elipse
    for $i = 0 \rightarrow contorno.tamanho$ **do**
        Calcula a distancia do ponto em $contorno[i]$ em relação à borda da elipse pela equação da elipse: $(contorno[i].x^2)/((largura/2)^2) + (contorno[i].y^2)/((largura/2)^2)$
        if $distancia < menorDistancia$ **then**
            $menorDistancia \leftarrow distancia$
            $ponto \leftarrow contorno[i]$
        end if
    end for
    Translada os pontos do $contorno$ para a sua posição original
    $c1 \leftarrow centro$
    $c2 \leftarrow centro$
    $c3 \leftarrow centro$
    Verifica a qual quadrante pertence o $ponto$ encontrado em $contorno$, e movimenta $c1$, $c2$ e $c3$ para as direções contrárias em x, y e (x, y) respectivamente
    $c1 \leftarrow$ retorno da chamada para o Algoritmo recursivo com ponteiro de $area1$
    $c2 \leftarrow$ retorno da chamada para o Algoritmo recursivo com ponteiro de $area2$
    $c3 \leftarrow$ retorno da chamada para o Algoritmo recursivo com ponteiro de $area3$
    Verifica qual dentre ar, $area1$, $area2$ e $area3$ tiveram aumento, atribui para o ponteiro $area$ e retorna o respectivo ponto $centro$, $c1$, $c2$ ou $c3$
else
    $area \leftarrow ar$
    return $centro$
end if

Fim.

Algoritmo 3 Projeta a Menor Elipse a um Ponto Central

Entrada: Lista de pontos de borda, $contorno = P1, \ldots, Pn$,
Ponto do centro a ser utilizado na iteração, $centro$,
Largura e Altura de referência para diminuição, $largura$, $altura$
Largura e Altura que serão modificadas para retornar à chamada anterior por ponteiro, $larguraAtual$, $alturaAtual$

Variáveis:

$distancia$: **real**
i, j, k : **inteiro**
t, a : **booleano**

Início

Translada os pontos do $contorno$ para a origem em relação ao centro do retângulo
$t \leftarrow a \leftarrow falso$
for $i = largura \rightarrow i < (largura/2)$ **do**
    $larguraAtual \leftarrow i$
    for $j = altura \rightarrow j < (altura/2)$ **do**
        $alturaAtual \leftarrow j$;
        for $k = 0 \rightarrow contorno.tamanho$ **do**
            Verifica se o ponto é interno pela equação da elipse :
            $distancia \leftarrow (contorno[k].x^2)/((i/2)^2) + (contorno[k].y^2)/((j/2)^2)$
            if $distancia \leqslant 1$ **then**
                $t \leftarrow verdadeiro$;
                Para Ciclo;
            end if
            if $k = (contorno.tamanho - 1)$ **and** $t = falso$ **then**
                $a \leftarrow verdadeiro$;
                Para Ciclo;
            end if
        end for
        Verifica se todos os pontos estiveram externos, indicado por a, parando o ciclo de diminuição da altura
        if $a = verdadeiro$ **then**
            Para Ciclo;
        end if
        $t \leftarrow falso$;
    end for
    Verifica se todos os pontos estiveram externos, indicado por a, parando o ciclo de diminuição da largura
    if $a = verdadeiro$ **then**
        Para Ciclo;
    end if
end for

Fim.

Algoritmo 4 Projeta a Maior Elipse a um Ponto Central

Entrada: Lista de pontos de borda, $contorno = P1, \ldots, Pn$,
Ponto do centro a ser utilizado na iteração, $centro$,
Largura e Altura de referência para aumento, $largura$, $altura$
Largura e Altura que serão modificadas para retornar à chamada anterior por ponteiro, $larguraAtual$, $alturaAtual$

Variáveis:

$distancia$: **real**
i, j, k : **inteiro**
t, a : **booleano**

Início

Translada os pontos do $contorno$ para a origem em relação ao centro do retângulo
$t \leftarrow a \leftarrow falso$
for $i = largura \rightarrow i > (largura + (largura/2))$ **do**
    $larguraAtual \leftarrow i$
    for $j = altura \rightarrow (altura + (altura/2))$ **do**
        $alturaAtual \leftarrow j$;
        for $k = 0 \rightarrow contorno.tamanho$ **do**
            Verifica se o ponto é externo pela equação da elipse :
            $distancia \leftarrow (contorno[k].x^2)/((i/2)^2) + (contorno[k].y^2)/((j/2)^2)$
            if $p \geqslant 1$ **then**
                $t \leftarrow verdadeiro$;
                Para Ciclo;
            end if
            if $k = (contorno.tamanho - 1)$ **and** $t = falso$ **then**
                $a \leftarrow verdadeiro$;
                Para Ciclo;
            end if
        end for
        Verifica se todos os pontos estiveram contidos, indicado por a, parando o ciclo de aumento da altura
        if $a = verdadeiro$ **then**
            Para Ciclo;
        end if
        $t \leftarrow falso$;
    end for
    Verifica se todos os pontos estiveram contidos, indicado por a, parando o ciclo de aumento da largura
    if $a = verdadeiro$ **then**
        Para Ciclo;
    end if
end for

Fim.

A partir dos Algoritmos 1, 2 e 3 apresentados, é possível realizar a projeção de uma elipse

completamente interna ao fitólito que busca otimizar seu tamanho recursivamente, porém, não garantindo o melhor caso. Como já afirmado, há um maior foco na elipse interna, pois todas suas medidas são levadas em conta, enquanto na elipse externa é necessário somente a sua área para fins de comparação de distância entre áreas. A heurística de se alcançar a distância entre as duas assume que estas sejam concêntricas, ou seja, possuem o mesmo ponto central, pois apenas desta forma é possível obter a informação necessária, que é a altura das protuberâncias do morfotipo. Para garantir esta informação, a busca da maior elipse externa aos pontos não se utiliza de heurística de otimização de tamanho, mantendo de forma fixa o ponto central da elipse interna já projetada.

Antes de realizar quaisquer extração de medidas, é preciso realizar uma ajuste de escala dos valores obtidos. Para tanto, foi aproveitada a barra de escala contida na imagem. O processo separação da barra de escala se deu pela utilização dos mesmos métodos utilizados na separação do morfotipo, com a diferença do uso de um valor de constante limiar maior na limiarização. Com a barra de escala isolada, uma busca em todos os pontos gerou a equação da reta entre dois pontos que tivessem a mesma largura, e a maior reta obtida foi entendida como o tamanho da barra de escala. Uma extração em imagens de exemplo mostrou que em geral a barra alcança a contagem de 97 pixels. Com isto, antes da apresentação dos dados extraídos do fitólito, é feita uma regra simples de três para normalizar os dados para uma barra de 97 pixels. O Algoritmo 5 é apresentado a seguir.

Além do Algoritmo 5, é necessário o cálculo do tamanho da barra de escala, mostrado em Algoritmo 6.

Algoritmo 5 Normalização dos dados

Entrada: altura da elipse interna, $altura$,
largura da elipse interna, $largura$,
tamanho da barra encontrada, $barra$,
distancia de áreas entre as elipses, $distancia$.
Variáveis:
1: $escala$: **real**;
Início
2: $escala \leftarrow 97$;
3: $largura \leftarrow (escala * largura)/barra$;
4: $altura \leftarrow (escala * altura)/barra$;
5: $distancia \leftarrow (escala * distancia)/barra$;
Fim.

Algoritmo 6 Calcula o tamanho da barra de escala

Entrada: Lista de pontos de borda, $contorno = P1, \ldots, Pn$.

Saída: Valor da maior reta que representa o tamanho da barra pela variável $maiorDistancia$.

Variáveis:

1: i, j : **inteiro**;
2: $maiorDistancia, x, y, distancia$: **real**;

Início

3: $maiorDistancia \leftarrow 0$;
4: **for** $i = 0 \rightarrow contorno.tamanho$ **do**
5: **for** $j = 0 \rightarrow contorno.tamanho$ **do**
6: **if** $contorno[i].x = contorno[j].x$ **then**
7: $x \leftarrow (contorno[j].x - contorno[i].x)^2$
8: $y \leftarrow (contorno[j].y - contorno[i].y)^2$;
9: $distancia \leftarrow \sqrt{(x + y)}$;
10: **if** $distancia > maiorDistancia$ **then**
11: $maiorDistancia \leftarrow distancia$;
12: **end if**
13: **end if**
14: **end for**
15: **end for**

Fim.

As chamadas para os Algoritmos apresentados devem realizar as medições esperadas, gerando o vetor de propriedades, que deve ser estruturado para a fase de análise e reconhecimento. O conjunto de dados de saída esperado segue desta forma:

<Largura>,<Altura>,<DiferençaDeÁreas>

que pode ser representado pelo exemplo:

61,5,5827.654785,Allagoptera Arenaria

Estes dados deverão ser manipulados pela última fase, que finaliza os passos do sistema de visão computacional, realizando a análise e interpretação destes dados. Os algoritmos em pseudocódigo apresentados nesta seção apenas mostram os objetivos e cálculos realizados pelo algoritmo original, porém, não seguem fielmente a ordem das chamadas e sua organização como um todo, o que é apresentado no Apêndice A.

Capítulo 5

Resultados e Discussão

Durante a seleção e preparação dos métodos abordados para as fases de pré-processamento e segmentação, foram realizados testes com várias imagens de morfotipos de fitólito, com objetivo de mapear quais técnicas melhor se adequariam ao tipo de imagem de entrada. Dentre os métodos de detecção de borda analisados, como afirmado no Capítulo 4, os de melhor resultado foram o método de Laplace, Canny e Suzuki.

Inicialmente, a implementação foi prevista para a linguagem de programação Java, visando a construção de um sistema de visão computacional completo a ser implantado no sistema proposto por [16], porém, a utilização da biblioteca OpenCV com a linguagem de programação C++ otimizou os processos e algoritmos, além de contribuir para a redução de problemas, que em conjunto com a biblioteca, a utilização do método proposto por [15] obteve o resultado mais otimizado, contribuindo diretamente para a construção dos algoritmos de extração de características, o que justifica a não utilização dos demais métodos.

A metodologia proposta neste trabalho possibilitou a extração de características de 120 imagens do banco de dados gerado pelo grupo de pesquisa citado no Capítulo 4, porém, foi possível identificar problemas e inconsistências nas fases de segmentação, detecção de bordas e extração de características, que serão apresentado nas Seções 5.1 e 5.2. A base de dados gerada segue no Apêndice B deste documento.

5.1 Fase de segmentação

A partir da aplicação dos métodos de limiarização em conjunto com a detecção de bordas, viu-se que a utilização do método proposto por [12] não se adequava à distribuição de tons de

cinza da imagem, apresentando resultados inesperados como mostra a Figura 5.1. Isto ocorre pois este método analisa os níveis de cores da imagem para decidir qual valor de limiar T deve ser utilizado, para então aplicar a limiarização fixa, e devido a presença da barra que é completamente branca, esta escolha acaba adotando valor T muito alto. Já a utilização da limiarização por um limiar fixo, diferente do método de Otsu, apresentou melhores resultados como mostra a Figura 5.2, onde foi possível identificar com maior facilidade o fitólito.

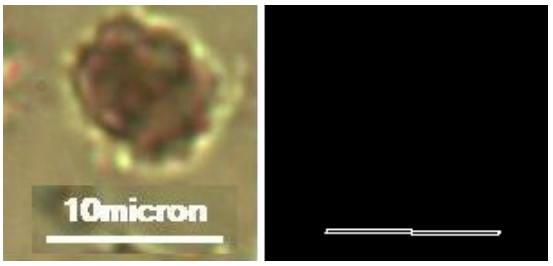

Figura 5.1: Processo de aplicação da limiarização por Otsu seguido da detecção de bordas por Suzuki num *globular echinate* da espécie *Allagoptera Arenaria*

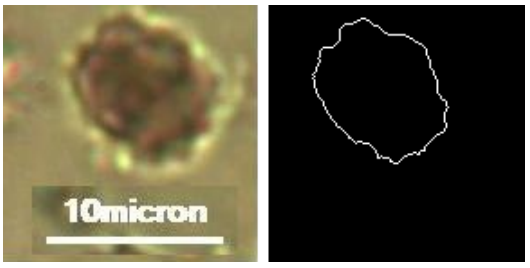

Figura 5.2: Processo de aplicação da limiarização por limiar fixo seguido da detecção de bordas por Suzuki num *globular echinate* da espécie *Allagoptera Arenaria*

Porém, há o problema da limiarização por limiar fixo, que é o fato de utilizar um fator constante, visto que cada imagem possui tamanho e distribuição de cores de forma diferenciada, apesar de se tratar do mesmo objeto de estudo. Este fato trouxe problemas no passo de segmentação, como apresentado nas Figuras 5.3 e 5.4. Os processos de aquisição da imagem também causam problemas à segmentação, por haver luminosidade e nitidez em padrões diferenciados entre as imagens, que ocorre também pelo estado do fitólito extraído, que pode se apresentar escuro ou com transparência.

Figura 5.3: Problemas apresentados no processo de aplicação da limiarização por limiar fixo seguido da detecção de bordas por Suzuki num *globular echinate* da espécie *Altalea Cuminis*

Figura 5.4: Problemas apresentados no processo de aplicação da limiarização por limiar fixo seguido da detecção de bordas por Suzuki num *globular echinate* da espécie *Altalea Cuminis*

Apesar dos problemas encontrados, o objetivo da fase de pré-processamento e segmentação é estático, devendo separar as bordas do objeto e gerar uma lista contendo os pontos deste objeto. Após, fez-se a implementação dos algoritmos e heurísticas apresentados no Capítulo 4, afim de gerar um programa completo para a extração de características, permitindo a análise do vetor de propriedades gerado por estes Algoritmos.

5.2 Fase de extração de características

Os algoritmos propostos no Capítulo 4 foram construídos a partir de heurísticas com o intuito de alcançar fielmente a extração de características tal como proposto por [2]. Como já explicado, esta extração visa a projeção de duas elipses, uma interna e outra externa, e partir destas três medições, extrair/calcular os eixos maior e menor da elipse interna e a diferença de áreas entre as elipses. A Figuras 5.5 e 5.6 mostram o resultado da aplicação dos algoritmos propostos.

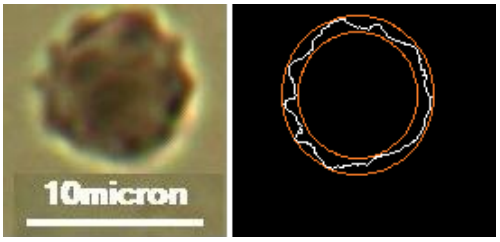

Figura 5.5: Aplicação dos algoritmos de extração de características para uma imagem da espécie *Poliandrococo Caudensis*

Figura 5.6: Aplicação dos algoritmos de extração de características para uma imagem da espécie *Bactris Bahienses*

A aplicação destes algoritmos foi desenvolvida esperando um bom resultado da fase anterior, ou seja, os percalços da segmentação acarretaram em mais problemas nesta fase. Os resultados apresentaram dois problemas, um proveniente da fase de segmentação, e outra do fato de a projeção da elipse interna não garantir o melhor caso.

O problema proveniente do mal resultado da fase de segmentação evidenciou a necessidade de otimização na qualidade do resultado dos métodos aplicados, como mostram as Figuras 5.7 e 5.8.

Figura 5.7: Problemas da aplicação dos algoritmos de extração de características por conta da fase de segmentação, para uma imagem da espécie *Astrocarium Aculialissinun*

Figura 5.8: Problemas da aplicação dos algoritmos de extração de características por conta da fase de segmentação, para uma imagem da espécie *Altalea Cuminis*

A heurística para a busca da menor elipse interna ao fitólito é a movimentação do ponto central da elipse na direção contrária em relação ao ponto de objeto mais próximo da elipse, que tende a levar esta elipse para uma região do fitólito com uma chance alta de obter um aumento em sua área. Utilizando a recursividade, a movimentação do ponto central é finalizada no momento em que não há mais aumento na área da elipse, e o valor de movimentação pode impedir que este ponto se mova para uma região onde ainda seja possível aumentar a sua área. As Figuras 5.9 e 5.10 apresentam este resultado.

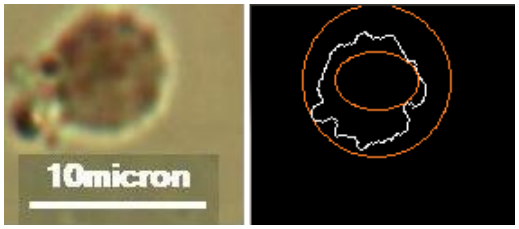

Figura 5.9: Problemas da aplicação dos algoritmos de extração de características devido à fase de extração de características, para uma imagem da espécie *Altalea Cuminis*

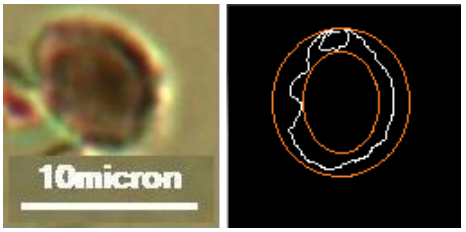

Figura 5.10: Problemas da aplicação dos algoritmos de extração de características devido à fase de extração de características, para uma imagem da espécie *Bactris Caryotaefolia*

5.3 Fase de Reconhecimento e Interpretação

A fase de reconhecimento e interpretação a partir da classificação consiste na definição de classes diferentes, que são as espécies, cujos dados relacionados são interpretados numa fase de treinamento, devendo reconhecer a distância entre estas classes para posterior identificação de dados de entrada cuja classe não é conhecida previamente.

A partir do Weka (*Waikato Environment for Knowledge Analysis*) [17] foi aplicado o método de classificação C4.5, que obteve uma taxa de acerto de 38%, que é um resultado insatisfatório, indicando que não houve diferença suficiente entre as classes. Não houve tempo hábil para melhor exploração desta fase, que foi realizada apenas como um teste preliminar.

Estes resultados apontam a necessidade de otimização dos processos em relação a qualidade destes dados, para que seja alcançada uma melhor diferenciação entre as classes.

A partir da aplicação dos métodos nas imagens obtidas, foi possível compreender o comportamento e a origem dos problemas que causaram inconsistências nos resultados. Uma melhor análise destes resultados e possibilidades de otimização são apresentados no Capítulo 6.

Capítulo 6

Considerações Finais

Durante o processo de testes utilizando métodos de limiarização, percebeu-se que cada imagem necessitava de uma constante limiar diferente para cada caso, o que dificultava a aplicação de apenas um limiar para todos os casos. Para resolver este problema, foi aplicada a limiarização pelo método de [12], porém, seu processo adaptativo acabava por ser influenciado pelos tons brancos da barra de escala da imagem, trazendo resultados incompatíveis. Também não foi possível a aplicação do Otsu de forma local, visto que não foi encontrado este tipo de implementação para testes.

Os problemas encontrados na fase de segmentação mostram claramente a partir dos fatos apresentados que a limiarização por uma constante limiar fixa não é suficiente para tratar todos os casos, e o estudo para a utilização de heurísticas que definam dinamicamente esta constante para este caso poderia comprometer o tempo de desenvolvimento.

Considerando o objetivo de construir uma ferramenta para uso acadêmico em pesquisas em outra área, a solução mais interessante seria permitir ao usuário desta ferramenta aplicar diferentes valores para a constante limiar em tempo de execução, deixando de responsabilidade do usuário a decisão do melhor valor a ser aplicado. Apesar de ser pouco arriscado o fato de permitir a decisão ao usuário, pode ser uma primeira tentativa de automatizar via aplicação os processos de análise de morfotipos de fitólito de *Arecaceae*.

Para a escolha das heurísticas de projeção da elipse na extração de características, evidenciou-se a dificuldade em projetar uma elipse interna a um conjunto de pontos e que seja ao mesmo tempo a maior possível. A primeira heurística utilizada foi a de buscar um ponto central inicial a partir do método *fitEllipse* disponível na biblioteca OpenCV, definida por [13], e projetar a elipse possuindo a maior largura possível, porém, quase sempre o ponto central

não era o ideal. A segunda heurística, que acabou por ser mantida na implementação, estende à heurística anterior, de projeção a partir do método mencionado, e seguido disso, repete esta projeção lançando novos pontos de centro em direções contrárias ao quadrante onde haja o choque mais próximo de um ponto com a borda da elipse. Esta heurística apresentou resultados bastante satisfatórios, porém, ainda não pôde garantir o melhor caso.

Além das duas heurísticas apresentadas, foi considerada a utilização de processos que utilizem o gradiente, partindo da ideia da segunda heurística, porém, lançando o ponto central para todas as direções possíveis ou para a metade com maior tendência de aumento. O uso do gradiente pode levar a um ciclo infinito, onde o ponto fique vagando numa mesma região passando pelos mesmos pontos, e ainda o tempo de execução para este método levaria muito mais tempo do que a heurística ora utilizada, que leva menos de um segundo para uma imagem de dimensões de aproximadamente $150x170$.

Além dos problemas em relação ao processamento de imagens, a base de dados gerada não é grande o suficiente para concluir sobre a eficácia, não só do método de classificação abordado, como também na forma como o vetor de propriedades é gerado.

Pensando novamente na construção da ferramenta para extração de características, poderia ser permitido novamente ao usuário a análise em tempo de execução dos resultados das projeções elipsais, podendo mover o centro dinamicamente caso o algoritmo não apresente um resultado satisfatório. Isto remete aos problemas já mencionados em relação ao usuário, porém, ainda assim pode ser uma solução local.

Em trabalhos futuros, pode-se realizar um estudo mais aprofundado de métodos matemáticos, como o dos quadrados mínimos, que apesar de apresentar o problema de tempo de execução, pode trazer com maior garantia o melhor caso, na tentativa de eliminar a responsabilidade do usuário. Além disto, as técnicas de limiarização também poderiam ser revisadas, buscando novos métodos que possam tratar as imagens para todos os casos.

Além dos fatores apresentados, a utilização de uma base de imagens maior possibilitaria a aplicação de testes utilizando métodos de classificação para os dados coletados de forma mais precisa, identificando as técnicas que melhor diferenciam as espécies, permitindo a construção completa de um sistema de visão computacional.

O fato que tornou o trabalho mais interessante foi a falta de trabalhos com aplicações com-

putacionais em imagens de morfotipos de fitólito, que trouxe o grande desafio não apenas de alcançar a extração de características, que são da área em específico, mas também nos processos de construção de uma ferramenta que auxilie no trabalho de reconhecimento dos morfotipos de fitólito. Certamente qualquer trabalho futuro que venha a complementar este, trará grandes avanços para as pesquisas relacionadas ao reconhecimento de morfotipos de fitólito, haja vista a falta de trabalhos deste sentido.

Apêndice A

Implementação dos Métodos e Algoritmos Propostos

```
#include <cv.h>
#include <highgui.h>
#include <math.h>

using namespace cv;

float findBar(Mat imageGray);
float barSize(vector<Point> contour);
vector<Point> rotatePoints(vector<Point> contour, Point C, float alfa);
vector<float> innerEllipseProjection(vector<Point> contour, RotatedRect
    rect);
Point recursaoEllipse(vector<Point> pts, Point centro, int a2, int b2,
    float areaAtual, float &area);
void smallerEllipse(vector<Point> pts, Point centro, int a2, int b2, int &w
    , int &h);
float externEllipseProjection(vector<Point> pts, float a2, float b2, Point
    centro, float ang, float ar);
void biggerEllipse(vector<Point> pts, Point centro, int a2, int b2, int &w,
     int &h);

int main(int argc, char** argv) {

  /*
  * Abre a imagem enviada por parâmetro em argv
  */
  Mat image;
  image = imread(argv[1], 1);

  if(argc != 2 || !image.data) {
    printf("No image data \n");
    return -1;
  }

  /*
  * Transforma para tons de cinza
  */
```

```
32  Mat gray;
33  cvtColor(image, gray, CV_RGB2GRAY);
34
35
36  /*
37  * Aplica a operação de Threshold
38  */
39  Mat otsu;
40  threshold(gray, otsu, 100, 230, THRESH_TOZERO_INV);
41
42  /*
43  * Encontra os contornos da imagem, selecionando o maior deles, que
        pertence ao objeto a ser medido
44  */
45  vector<vector<Point> > contours;
46  vector<Vec4i> hierarchy;
47  findContours(otsu, contours, hierarchy, CV_RETR_TREE,
        CV_CHAIN_APPROX_NONE, Point(0,0));
48
49  int index = 0, nu = 0;
50  for(int i = 0; i<contours.size();i++){
51    if(contours[i].size() > nu){
52      index = i;
53      nu = contours[i].size();
54    }
55  }
56
57  /*
58  * Encontra o melhor circulo para o objeto da imagem
59  */
60  RotatedRect minEllipse;
61  minEllipse = fitEllipse( Mat(contours[index]) );
62
63  /*
64  * Realiza o cálculo do tamanho da barra de escala presente na imagem
65  */
66  float bar = findBar(gray);
67
68  /*
69  * Método para a melhor elipse interna, que chama em seguida o método da
        elipse externa
70  * O vetor de retorno da função contem em ordem: Lado maior interno; Lado
        menor interno; Diferença entre áreas
71  */
72  vector<float> measure = innerEllipseProjection(contours[index],
        minEllipse);
73
74  int w = measure[0];
75  int h = measure[1];
76  float dist = measure[2];
77
78  //Regra de escala para padrao 97
79  int escala = 97;
80  w = (escala * w) / bar;
81  h = (escala * h) / bar;
```

```
82    dist = (escala * dist) / bar;
83
84    printf("%d,%d,%f\n", w, h, dist);
85
86    return 0;
87  }
88
89  float findBar(Mat imageGray) {
90
91    Mat img;
92    threshold(imageGray, img, 210, 10, THRESH_TOZERO);
93
94    vector<vector<Point> > contours;
95    vector<Vec4i> hierarchy;
96    findContours(img, contours, hierarchy, CV_RETR_TREE, CV_CHAIN_APPROX_NONE
          , Point(0,0));
97
98    int index = 0, nu = 0;
99    for(int i = 0; i<contours.size();i++){
100     if(contours[i].size() > nu){
101       index = i;
102       nu = contours[i].size();
103     }
104   }
105
106   float ret = barSize(contours[index]);
107   return ret;
108
109 }
110
111   /*
112   * Este método encontra a maior reta interna possível
113   */
114 float barSize(vector<Point> contour) {
115   int it = contour.size();
116   float mdist = 0;
117
118   for(int i = 0; i < it; i++) {
119     for(int j = 0; j < it; j++) {
120       if(contour[i].y == contour[j].y) {
121         float x = pow((contour[j].x - contour[i].x), 2);
122         float y = pow((contour[j].y - contour[i].y), 2);
123         float d = sqrt((x+y));
124         if(d > mdist) {
125           mdist = d;
126         }
127       }
128     }
129   }
130
131   return mdist;
132 }
133
134
135 vector<Point> rotatePoints(vector<Point> contour, Point C, float alfa) {
```

```
136   vector<Point> aux = contour;
137   for(int i = 0; i < contour.size(); i++) {
138     aux[i].x -= C.x;
139     aux[i].y -= C.y;
140     contour[i].x = (aux[i].x * cos(alfa)) - (aux[i].y * sin(alfa));
141     contour[i].y = (aux[i].x * sin(alfa)) + (aux[i].y * cos(alfa));
142     contour[i].x += C.x;
143     contour[i].y += C.y;
144     aux[i].x += C.x;
145     aux[i].y += C.y;
146   }
147   return contour;
148 }
149
150 vector<float> innerEllipseProjection(vector<Point> contour, RotatedRect
      rect, Mat *img, vector<Vec4i> hierarchy) {
151
152   Point centro = rect.center;
153   float b2, a2;
154   if(rect.size.width > rect.size.height) {
155     a2 = rect.size.width;
156     b2 = rect.size.height;
157   } else {
158     b2 = rect.size.width;
159     a2 = rect.size.height;
160   }
161
162   vector<Point> quad (2);
163   quad[0] = Point(centro.x+(rect.size.width/2), centro.y-(rect.size.height
        /2));
164   quad[1] = Point(centro.x+(rect.size.width/2), centro.y+(rect.size.height
        /2));
165   quad = rotatePoints(quad, centro, (rect.angle * M_PI / 180));
166   float tg = (float)(quad[0].y-quad[1].y)/(float)(quad[0].x-quad[1].x);
167   float ang = -atan(tg);
168   vector<Point> pts = rotatePoints(contour, centro, -ang);
169   float angle = rect.angle * M_PI / 180;
170
171   int w, h;
172   smallerEllipse(pts, centro, a2, b2, w, h);
173   float area = M_PI * h * w;
174
175     float menor = 2000;
176     Point po;
177
178     for(int i = 0; i < pts.size(); i++) {
179       pts[i].x -= centro.x;
180       pts[i].y -= centro.y;
181     }
182
183     float p;
184     for(int k = 0; k < pts.size(); k++) {
185       p = (pow(pts[k].x, 2)/pow((w/2), 2)) + (pow(pts[k].y, 2)/pow((h/2),
            2));
186       if (p<menor){
```

```
187           menor = p;
188           po = pts[k];
189         }
190       }
191
192       for(int i = 0; i < pts.size(); i++) {
193         pts[i].x += centro.x;
194         pts[i].y += centro.y;
195       }
196
197       Point c1 = centro, c2 = centro, c3 = centro, c4 = centro;
198
199       int razao = 10;
200
201       //Primeiro quadrante
202       if(po.x > 0 && po.y > 0) {
203         c1.x -= razao;
204         c2.y -= razao;
205       }
206       //Segundo quadrante
207       if(po.x < 0 && po.y > 0) {
208         c1.x += razao;
209         c2.y -= razao;
210       }
211       //Terceiro quadrante
212       if(po.x < 0 && po.y < 0) {
213         c1.x += razao;
214         c2.y += razao;
215       }
216       //Quarto quadrante
217       if(po.x > 0 && po.y < 0) {
218         c1.x -= razao;
219         c2.y += razao;
220       }
221
222       float ar1, ar2, ar3;
223       c1 = recursaoEllipse(pts, c1, a2, b2, area, ar1);
224       c2 = recursaoEllipse(pts, c2, a2, b2, area, ar2);
225
226       if (area >= ar1 && area >= ar2){
227       }else if(ar1 >= ar2) {
228           area = ar1;
229           centro = c1;
230         } else {
231         area = ar2;
232         centro = c2;
233       }
234
235     smallerEllipse(pts, centro, a2, b2, w, h);
236     area = M_PI * h * w;
237     float dist = externEllipseProjection(pts, a2, b2, centro, -ang, area, img
        , hierarchy);
238
239     vector<float> ret (3);
240     ret[0] = w;
```

```
241   ret[1] = h;
242   ret[2] = dist;
243
244   return ret;
245 }
246
247 Point recursaoEllipse(vector<Point> pts, Point centro, int a2, int b2,
      float areaAtual, float &area) {
248   int w, h;
249   smallerEllipse(pts, centro, a2, b2, w, h);
250   float ar = M_PI * h * w;
251
252   //Recursão
253   if(ar > areaAtual) {
254     float menor = 2000;
255     Point po;
256
257     for(int i = 0; i < pts.size(); i++) {
258       pts[i].x -= centro.x;
259       pts[i].y -= centro.y;
260     }
261
262     float p;
263     for(int k = 0; k < pts.size(); k++) {
264       p = (pow(pts[k].x, 2)/pow((w/2), 2)) + (pow(pts[k].y, 2)/pow((h/2),
            2));
265       if (p<menor){
266         menor = p;
267         po = pts[k];
268       }
269     }
270
271     for(int i = 0; i < pts.size(); i++) {
272       pts[i].x += centro.x;
273       pts[i].y += centro.y;
274     }
275
276     Point c1 = centro, c2 = centro, c3 = centro;
277     int razao = 5;
278
279     //Primeiro quadrante
280     if(po.x > 0 && po.y > 0) {
281       c1.x -= razao;
282       c2.y -= razao;
283       c3.x -= razao;
284       c3.y -= razao;
285     }
286     //Segundo quadrante
287     if(po.x < 0 && po.y > 0) {
288       c1.x += razao;
289       c2.y -= razao;
290       c3.x += razao;
291       c3.y -= razao;
292     }
293     //Terceiro quadrante
```

```
294     if(po.x < 0 && po.y < 0) {
295       c1.x += razao;
296       c2.y += razao;
297       c3.x += razao;
298       c3.y += razao;
299     }
300     //Quarto quadrante
301     if(po.x > 0 && po.y < 0) {
302       c1.x -= razao;
303       c2.y += razao;
304       c3.x -= razao;
305       c3.y += razao;
306     }
307
308     float ar1, ar2, ar3;
309     c1 = recursaoEllipse(pts, c1, a2, b2, ar, ar1);
310     c2 = recursaoEllipse(pts, c2, a2, b2, ar, ar2);
311     c3 = recursaoEllipse(pts, c3, a2, b2, ar, ar3);
312
313     if (ar >= ar1 && ar >= ar2 && ar >= ar3){
314       area = ar;
315       return centro;
316     } else if(ar1 >= ar2 && ar1 >= ar3) {
317       area = ar1;
318       return c1;
319     } else if(ar2 >= ar3) {
320       area = ar2;
321       return c2;
322     } else {
323       area = ar3;
324       return c3;
325     }
326
327   } else {
328     area = ar;
329     return centro;
330   }
331 }
332
333 void smallerEllipse(vector<Point> pts, Point centro, int a2, int b2, int &w
      , int &h) {
334
335   for(int i = 0; i < pts.size(); i++) {
336     pts[i].x -= centro.x;
337     pts[i].y -= centro.y;
338   }
339
340   bool t = false, a = false;
341   for(int i = a2; i > a2/2; i--) {
342     w = i;
343     for(int j = b2; j > b2/2; j--) {
344       h = j;
345       for(int k = 0; k < pts.size(); k++) {
346         float p = (pow(pts[k].x, 2)/pow((i/2), 2)) + (pow(pts[k].y, 2)/pow
              ((j/2), 2));
```

```
347           if(p <= 1) {
348             t = true;
349             break;
350           }
351           if(k == pts.size()-1 && !t) {
352             a = true;
353             break;
354           }
355         }
356         if(a) {
357           break;
358         }
359         t = false;
360       }
361       if(a) {
362         break;
363       }
364     }
365 }
366
367 float externEllipseProjection(vector<Point> pts, float a2, float b2, Point
      centro, float ang, float ar, Mat *img, vector<Vec4i> hierarchy) {
368   int w, h;
369   biggerEllipse(pts, centro, a2, b2, w, h);
370   float area = M_PI * h * w;
371   area = area - ar;
372   return area;
373 }
374
375 void biggerEllipse(vector<Point> pts, Point centro, int a2, int b2, int &w,
      int &h) {
376   for(int i = 0; i < pts.size(); i++) {
377     pts[i].x -= centro.x;
378     pts[i].y -= centro.y;
379   }
380
381   bool t = false, a = false;
382   for(int i = a2; i < a2+(a2/2); i++) {
383     w = i;
384     for(int j = b2; j < b2+(b2/2); j++) {
385       h = j;
386       for(int k = 0; k < pts.size(); k++) {
387         float p = (pow(pts[k].x, 2)/pow((i/2), 2)) + (pow(pts[k].y, 2)/pow
              ((j/2), 2));
388         if(p >= 1) {
389           t = true;
390           break;
391         }
392         if(k == pts.size()-1 && !t) {
393           a = true;
394           break;
395         }
396       }
397       if(a) {
398         break;
```

```
399             }
400             t = false;
401         }
402         if(a) {
403             break;
404         }
405     }
406 }
```

Apêndice B

Base de Dados da Extração de Características

Estrutura:

<Eixo_Maior>,<Eixo_Menor>,<Distância>,<Espécie>

Dados:

61,5,5827.654785,Allagoptera-Arenaria
59,59,10948.450195,Allagoptera-Arenaria
30,21,14146.591797,Allagoptera-Arenaria
109,49,30746.765625,Allagoptera-Arenaria
71,51,33850.660156,Allagoptera-Arenaria
79,71,15013.671875,Allagoptera-Arenaria
43,59,22741.986328,Allagoptera-Arenaria
65,37,14598.982422,Allagoptera-Arenaria
61,45,29942.521484,Allagoptera-Arenaria
45,51,17922.787109,Allagoptera-Arenaria
55,51,40605.085938,Allagoptera-Arenaria
75,43,31412.785156,Allagoptera-Arenaria
67,75,18595.085938,Allagoptera-Arenaria
85,37,17564.644531,Allagoptera-Arenaria
79,55,17715.441406,Allagoptera-Arenaria
83,77,42477.476562,Allagoptera-Arenaria
81,41,23232.078125,Allagoptera-Arenaria
43,27,11633.318359,Allagoptera-Arenaria
63,30,44315.308594,Altalea-Cuminis
43,27,14950.839844,Altalea-Cuminis
36,26,21563.890625,Altalea-Cuminis
40,35,21589.023438,Altalea-Cuminis
41,33,23354.599609,Altalea-Cuminis
21,19,6889.512695,Altalea-Cuminis
30,24,11422.830078,Altalea-Cuminis
26,21,8413.185547,Altalea-Cuminis
43,45,14504.732422,Altalea-Cuminis

33,37,11507.653320,Altalea-Cuminis
41,33,20907.298828,Altalea-Cuminis
49,49,18092.433594,Altalea-Cuminis
21,19,5394.114258,Astrocarium-Aculialissinun
49,41,5752.256348,Astrocarium-Aculialissinun
16,8,1985.486450,Astrocarium-Aculialissinun
20,12,5155.353516,Astrocarium-Aculialissinun
19,10,3898.716553,Astrocarium-Aculialissinun
25,15,3345.796387,Astrocarium-Aculialissinun
31,17,4413.937500,Astrocarium-Aculialissinun
43,23,8152.432129,Astrocarium-Aculialissinun
39,31,10728.539062,Astrocarium-Aculialissinun
33,51,7178.539062,Astrocarium-Aculialissinun
31,33,11564.202148,Bactris-Bahienses
30,28,13446.016602,Bactris-Bahienses
51,25,7103.141602,Bactris-Bahienses
57,47,4439.070312,Bactris-Bahienses
57,45,18331.193359,Bactris-Bahienses
47,33,19405.617188,Bactris-Bahienses
67,55,22289.597656,Bactris-Bahienses
81,57,15026.237305,Bactris-Bahienses
65,63,7593.228516,Bactris-Bahienses
67,61,8799.601562,Bactris-Bahienses
49,51,22811.103516,Bactris-Bahienses
47,33,16766.679688,Bactris-Bahienses
53,49,11947.476562,Bactris-Bahienses
57,45,9459.335938,Bactris-Bahienses
47,30,32854.777344,Bactris-Bahienses
55,37,10320.132812,Bactris-Bahienses
36,26,21469.644531,Bactris-Caryotaefolia
29,21,4721.813477,Bactris-Caryotaefolia
49,41,12801.989258,Bactris-Caryotaefolia
41,31,15585.442383,Bactris-Caryotaefolia
61,55,14090.042969,Bactris-Caryotaefolia
29,37,4168.893555,Bactris-Caryotaefolia
43,56,12013.451172,Bactris-Caryotaefolia
32,29,15117.343750,Bactris-Caryotaefolia
31,27,5852.787598,Bactris-Caryotaefolia
45,27,6701.016602,Euterpe-Edulis
109,81,94854.117188,Euterpe-Edulis
47,33,20574.289062,Euterpe-Edulis
27,29,10360.972656,Euterpe-Edulis
165,155,78693.750000,Euterpe-Edulis
36,24,19716.634766,Euterpe-Edulis
133,125,72404.289062,Euterpe-Edulis

103,107,57060.753906,Euterpe-Edulis
251,145,188438.984375,Euterpe-Edulis
217,147,152156.750000,Euterpe-Edulis
54,32,40733.890625,Geonoma-Elegans
75,47,44318.449219,Geonoma-Elegans
65,49,17087.123047,Geonoma-Elegans
66,35,54299.289062,Geonoma-Elegans
59,31,16138.360352,Geonoma-Elegans
61,71,37815.351562,Geonoma-Elegans
55,59,28248.250000,Geonoma-Elegans
97,59,27762.255859,Geonoma-Elegans
93,45,26939.156250,Geonoma-Elegans
73,51,27598.892578,Geonoma-Elegans
49,43,27479.513672,Geonoma-Elegans
81,79,23596.505859,Geonoma-Elegans
79,47,30225.263672,Geonoma-Elegans
69,51,17043.140625,Geonoma-Elegans
63,39,38022.695312,Geonoma-Elegans
69,53,11180.927734,Geonoma-Elegans
61,61,44180.214844,Geonoma-Elegans
73,51,28327.742188,Geonoma-Elegans
83,51,51732.609375,Geonoma-Elegans
71,43,55854.375000,Geonoma-Elegans
55,55,43652.429688,Geonoma-Elegans
57,59,41883.714844,Geonoma-Elegans
45,35,31463.050781,Geonoma-Elegans
57,53,43099.511719,Geonoma-Elegans
83,47,25644.820312,Geonoma-Elegans
87,73,81983.414062,Geonoma-Rodienses
57,39,30134.310547,Geonoma-Rodienses
61,35,32247.791016,Geonoma-Rodienses
61,65,8168.850098,Geonoma-Rodienses
76,54,10709.941406,Geonoma-Rodienses
37,49,12337.035156,Poliandrococo-Caudensis
51,39,14335.085938,Poliandrococo-Caudensis
51,53,22383.847656,Poliandrococo-Caudensis
67,71,8278.095703,Poliandrococo-Caudensis
67,71,8278.095703,Poliandrococo-Caudensis
89,47,16389.687500,Poliandrococo-Caudensis
71,67,40925.523438,Poliandrococo-Caudensis
65,39,21567.033203,Poliandrococo-Caudensis
37,49,17376.150391,Poliandrococo-Caudensis
43,37,6663.317871,Poliandrococo-Caudensis
33,27,11149.511719,Poliandrococo-Caudensis
51,31,14096.326172,Poliandrococo-Caudensis

65,41,12739.157227,Poliandrococo-Caudensis
59,31,12852.256836,Poliandrococo-Caudensis
125,79,56589.511719,Poliandrococo-Caudensis

Referências Bibliográficas

[1] ROVNER, I. Potential of opal phytoliths for use in palaeoecological reconstruction. *Quaternary Research*, San Diego, v. 1, n. 3, p. 345–359, 1971.

[2] ALBERT, R. M.; BAMFORD, M. K.; CABANES, D. Palaeoecological significance of palms at olduvai gorge, tanzania, based on phytolith remains. *Quatern. Int.*, v. 193, n. 1-2, p. 41–48, 2009.

[3] BREMOND, L. et al. Grass water stress estimated from phytoliths in West Africa. *Journal of Biogeography*, Blackwell Science Ltd, v. 32, n. 2, p. 311–327, fev. 2005. ISSN 0305-0270. Disponível em: <http://dx.doi.org/10.1111/j.1365-2699.2004.01162.x>.

[4] PIPERNO, D. R.; BECKER, P. Vegetational history of a site in the central amazon basin derived from phytolith and charcoal records from natural soils. *Quaternary Research*, v. 45, n. 2, p. 202–209, 1996.

[5] CALEGARI, M. R. et al. Opal phytolith extraction in oxisols. *Quaternary International*, 2011. Disponível em: <http://dx.doi.org/10.1016/j.quaint.2011.11.005>.

[6] PEDRINI, H.; SCHWARTZ, W. R. *Análise de Imagens Digitais: Princípios, Algoritmos e Aplicações*. [S.l.]: Editora Thomson Learning, 2007. 528 p. ISBN 9788522105953.

[7] GONZALEZ, R. C.; WOODS, R. E. *Processamento de Imagens Digitais*. [S.l.]: Edgard Blucher, 2000.

[8] RUNGE, F. The opal phytolith inventory of soils in central africa - quantities, shapes, classification, and spectra. *Review of Palaeobotany and Palynology*, v. 107, n. 1-2, p. 23–53, 1999.

[9] BALL, T. B.; EHLERS, R.; STANDING, M. D. Review of typologic and morphometric analysis of phytoliths produced by wheat and barley. *Breeding science*, Japanese Society of Breeding, v. 59, n. 5, p. 505–512, 2009. ISSN 13447610.

[10] SENDULSKY, T.; LABOURIAU, L. G. Corpos silicosos de gramíneas dos cerrados. *Anais da Academia Brasileira de Ciências*, São Paulo, p. 159–170, 1966.

[11] CRISTOBAL, R. M. A. Nuevo sistema de análisis descriptivo para fitolitos de sílice. *Pyrenae*, n. 26, p. 19–38, 1995. ISSN 00798215. Spa.

[12] OTSU, N. A threshold method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-9, n. 1, p. 62–66, 1979.

[13] BRADSKI, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[14] CANNY, J. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8, n. 6, November 1986.

[15] SUZUKI, S.; BE, K. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, v. 30, n. 1, p. 32–46, abr. 1985. ISSN 0734189X.

[16] SOUZA, L. G.; BOSCARIOLI, C.; CALEGARI, M. R. Levantamento de requisitos para software web voltado à classificação de morfotipos de fitólitos. *Anais Eletrônicos do XX Encontro Anual De Iniciação Científica*, Ponta Grossa, 2011. Disponível em: <http://www.eventos.uepg.br/eaic/anais/>.

[17] HALL, M. et al. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, ACM, New York, NY, USA, v. 11, n. 1, p. 10–18, nov. 2009. ISSN 1931-0145.