UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
COORDENADORIA DO CURSO DE ENGENHARIA DE SOFTWARE

FELIPE REIS PAVAN

APLICATIVO MÓVEL DE ANÁLISE DE TEXTURA BASEADO EM FRACTAIS

TRABALHO DE CONCLUSÃO DE CURSO

DOIS VIZINHOS

2018

FELIPE REIS PAVAN

APLICATIVO MÓVEL DE ANÁLISE DE TEXTURA BASEADO EM FRACTAIS

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Engenharia de Software, da Universidade Tecnológica Federal do Paraná.

Orientador: Professor Andre Luiz Marasca

DOIS VIZINHOS

2018

TERMO DE APROVAÇÃO

Aplicativo móvel de análise de textura baseado em fractais

por

Felipe Reis Pavan

Este Trabalho de Conclusão de Curso foi apresentado em 04 de Dezembro de 2018 como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software. O(a) candidato(a) foi arguido(a) pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Andre Luiz Marasca
Presidente da Banca

Andre Roberto Ortoncelli
Membro Titular

Marlon Marcon
Membro Titular

* A Folha de Aprovação assinada encontra-se na Coordenação do Curso

Dedico este trabalho a Deus, minha família e minha namorada.

AGRADECIMENTOS

Primeiramente, agradeço a Deus por me permitir alcançar esta conquista tão desejada e esperada. Em todos os momentos e orações ele foi meu porto seguro.

Agradeço à minha família. Aos meus pais, que durante todo curso me incentivaram a continuar nos momentos nos quais eu já pensava em desistir, além disso deram total apoio e suporte para que este trabalho fosse realizado. Ao meu irmão, companheiro de todos os dias e companheiro em todo o curso, também faz parte desta conquista.

À minha namorada (se Deus quiser minha futura esposa), que além de todo amor e carinho por mim, sempre me incentivou, apoiou e em muitos momentos me aconselhou. Sou realizado e muito feliz por tê-la em minha vida.

Também quero agradecer ao professor Andre Luiz Marasca, que de maneira muito compreensiva e prestativa me auxiliou no desenvolvimento deste trabalho.

Por fim, externo este agradecimento aos meu colegas, aos meus amigos e a todos que de alguma forma contribuíram para o desenvolvimento deste trabalho.

Prefiram a minha instrução à prata, e o conhecimento ao ouro puro, pois a sabedoria é mais preciosa do que rubis; nada do que vocês possam desejar compara-se a ela. (Provérbios 8:10-11)

RESUMO

PAVAN, Felipe. APLICATIVO MÓVEL DE ANÁLISE DE TEXTURA BASEADO EM FRACTAIS. 63 f. Trabalho de Conclusão de Curso – Coordenadoria do curso de Engenharia de Software, Universidade Tecnológica Federal do Paraná. Dois Vizinhos, 2018.

Este trabalho propõe desenvolver um aplicativo móvel capaz de classificar imagens de textura, baseando-se na teoria fractal. A proposta consiste em desenvolver um aplicativo móvel, para Android, que receba um modelo treinado, capture a imagem de uma textura, extraia as características utilizando o recente método de análise de textura baseado em descritores fractais denominado FD_{MIC} e então classifique-a. O modelo de classificação treinado é gerado por meio de um software *desktop*, desenvolvido em Java, que recebe parâmetros de entrada como uma base de imagens, o valor de escala e o tamanho da subimagem a ser cortada. Esta proposta visa unir a performance do FD_{MIC} à mobilidade proporcionada por um *smartphone*. Por fim, por meio de testes experimentais, buscou-se medir a acurácia do método FD_{MIC} nesta solução proposta e, por meio de análises estatísticas, comparou-se os resultados, enfatizando pontos fortes e fracos desta abordagem.

Palavras-chave: Análise de textura. Dimensão Fractal. Descritores Fractais. Aplicativo de análise de textura. Aplicativo Móvel.

ABSTRACT

PAVAN, Felipe. APLICATIVO MÓVEL DE ANÁLISE DE TEXTURA BASEADO EM FRACTAIS. 63 f. Trabalho de Conclusão de Curso – Coordenadoria do curso de Engenharia de Software, Universidade Tecnológica Federal do Paraná. Dois Vizinhos, 2018.

This work proposes to develop a mobile application to classify texture images, based on fractal theory. The proposal is to develop an Android mobile application that receives a trained model, captures images of a texture, extracts the features using a recent texture analysis method based on the fractal descriptors called FD_{MIC} and then classify it. The trained classification model is generated through a desktop software, developed in Java, which receives input parameters such as an image base, the scale value and the size of the sub-image to be cut. This proposal aims to unite the performance of FD_{MIC} to the mobility provided by a smartphone. Finally, after experimental tests, the accuracy of FD_{MIC} in this proposed solution was measured and through statistical analyzes, the results were compared, emphasizing strengths and weaknesses of this approach.

Keywords: Texture analysis. Fractal Dimension. Fractional Descriptors. Texture analysis application. Mobile application.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Há anos a computação tenta imitar capacidades humanas como andar, falar, pensar e ver, que comumente então interligadas. Na computação, existe uma área denominada Visão Computacional que lida com tarefas unindo a capacidade de ver e pensar. Ela extrai informações de imagens de forma automatizada, utilizando abordagens estatísticas ou geométricas, para resolver problemas reais (SOLEM, 2012).

Dentro da visão computacional, existe a subárea de análise de textura que tornou-se objeto de diversos estudos, impulsionada, principalmente, pelos avanços da inteligência artificial e da capacidade de processamento computacional nas últimas décadas.

A textura, segundo Gunasekara et al. (2017), pode ser definida como uma propriedade da superfície de um objeto ou região, que contém importantes informações sobre a sua estrutura e padrão. Já a análise de textura pode ser entendida como a caracterização das regiões em uma imagem a partir do conteúdo da textura (MATHWORKS, 2018) ou como a quantificação das características de uma textura baseado em um vetor de características (GUNASEKARA et al., 2017).

Dessa forma, a análise de textura extrai características das imagens de textura de modo a obter assinaturas similares entre imagens que pertencem à mesma classe e assinaturas não similares entre classes diferentes. Isto permite distinguir imagens de classes distintas quando aplicadas à um algoritmo de aprendizagem de máquina.

Exemplos de aplicações podem ser vistos na área da medicina, como auxílio no diagnóstico de imagens do intestino delgado, da retina e mamografias (KHADEMI; KRISHNAN, 2008) e na detecção de doenças de pele (ISLAM et al., 2017). Na área de segurança, como identificação pessoal a partir do reconhecimento da íris (MA et al., 2003). Áreas biológicas, a agricultura, indústria de manufatura, nanotecnologia e setor alimentício também se beneficiam deste ramo da visão computacional.

Porém, apesar da crescente empregabilidade da visão computacional, ainda há processos e atividades realizadas por especialistas humanos. No entanto, é um processo custoso

encontrar, manter e treinar um especialista. Além disso, eles sofrem com fatores externos, tais como cansaço e estresse (MALAMAS et al., 2003), que podem prejudicar o resultado final. Por outro lado, um sistema computacional automatizado pode fazer o mesmo trabalho, de modo mais rápido e mais preciso (PRASAD et al., 2011).

Em sua maior parte, sistemas automatizados para classificação de imagens são implementados em computadores de grande porte, robustos e estáticos, ou seja, não permitem a mobilidade. Isso dificulta o trabalho de profissionais que podem se beneficiar da análise de textura, como biólogos, agrônomos, agricultores e até mesmo os próprios estudantes destas áreas.

A exemplo disso, considere um estudante do curso de ciências biológicas ou qualquer leigo, que deseja mapear as espécies vegetais de uma determinada região. Um software de análise de textura implementado em um dispositivo móvel pode ajudá-lo a classificar cada espécie, mesmo estando em campo.

1.1 PROBLEMA

Como visto anteriormente, existem diversas aplicações da visão computacional, no segmento de análise de textura, para solucionar problemas específicos em diversas áreas. Estes softwares realizam o processamento completo da análise de textura, envolvendo fases desde a aprendizagem até a classificação. Isso exige um hardware de processamento robusto, dificultando e inviabilizando o uso desta tecnologia em dispositivos móveis, como *smartphones*.

No entanto, o processo de classificação isolado, que consiste na extração de características e classificação supervisionada, não é um trabalho tão custoso para estes dispositivos. Neste sentido, propõe-se a desenvolver uma solução de análise de textura para dispositivos móveis. Para tanto, pretende-se utilizar um método recente de análise de textura baseado na teoria fractal (CASANOVA et al., 2016; MARASCA, 2016), conhecido como FD_{MIC} (*Fractal descriptors estimated by the mutual interference of color channels* ou descritores fractais estimados pela interferência mútua de canais de cores).

1.2 OBJETIVOS

1.2.1 OBJETIVO GERAL

Desenvolver aplicativo móvel capaz de realizar classificação de imagens por meio da análise de textura baseada na teoria fractal.

1.2.2 OBJETIVOS ESPECÍFICOS

- Implementar extrator de características FD_{MIC} na linguagem Java.
- Verificar a acurácia do método FD_{MIC} utilizando benchmarks e validação cruzada.
- Construir uma base de imagens para o estudo de caso.
- Verificar a acurácia do método FD_{MIC} sobre a base de imagens do estudo de caso.
- Desenvolver software para gerar o modelo classificador para o estudo de caso.
- Desenvolver aplicativo móvel capaz de receber um modelo treinado e realizar a classificação das imagens.
- Integrar modelo com o aplicativo criado.

1.3 JUSTIFICATIVA

Atualmente, existem inúmeras aplicações de visão computacional, em diversos segmentos, sendo utilizadas em processos que antes dependiam de monitoramento e inspeção manual. Estas aplicações reduziram tempo de execução e custos com mão de obra e, melhoraram resultados e performance considerando o contexto que estão inseridas.

Neste cenário, a textura desempenha um papel importante, pois é capaz de extrair características ricas e exclusivas de um objeto, permitindo descrevê-lo e identificá-lo. Exemplos de aplicações que empregam a análise de textura podem ser encontrados em diagnósticos médicos e na inspeção de qualidade na indústria.

Contudo, grande parte destas aplicações realizam o processo completo de análise (extração de características, aprendizado supervisionado e etapa de classificação), exigindo um considerável poder de processamento, sendo uma característica de dispositivos mais robustos, restringindo a sua aplicabilidade à um contexto mais específico.

Neste sentido, o presente trabalho de desenvolvimento de uma aplicação móvel, busca unir dois pontos importantes: a performance de uma análise de textura à mobilidade proporcionada por um *smartphone*. Somando-se a capacidade da aplicação executar em locais onde não é possível a conexão com a internet.

Além disso, o método de análise de textura FD_{MIC} é recente e foi escolhido baseado na sua elevada acurácia de classificação. Como estudo de caso, a aplicação explorou o problema de classificação de plantas nativas na UTFPR, câmpus Dois Vizinhos-PR, tornando este trabalho

multidisciplinar, além de ser uma tentativa de desenvolver uma ferramenta capaz de ser utilizada por alunos ou leigos no assunto.

1.4 ESTRUTURA DO TRABALHO

Este trabalho está estruturado da seguinte forma: o Capítulo 2 realiza uma fundamentação teórica dos tópicos referentes ao projeto proposto. O Capítulo 3 apresenta exemplos de aplicações no mercado que utilizam da análise de textura para solucionar problemas. O Capítulo 4 descreve os materiais e métodos utilizados para alcançar os objetivos propostos. O Capítulo 5 apresenta os resultados das interfaces das aplicações desenvolvidas. O Capítulo 6 apresenta os testes realizados e os resultados obtidos. Por fim, o Capítulo 7 tece considerações finais, bem como as contribuições e os trabalhos futuros identificados por meio deste projeto.

2 REFERENCIAL TEÓRICO

Neste capítulo são introduzidos conceitos necessários para a compreensão da análise de textura desde do método de extração de características baseado em fractais (FD_{MIC}) até a classificação supervisionada por meio do método *Linear Discriminant Analysis* (LDA) e o método *k-fold* para validação dos resultados.

2.1 ANÁLISE DE TEXTURA

Em linhas gerais, a análise de textura possibilita extrair características, diretamente relacionadas com as propriedades físicas, da superfície de algum objeto. Conforme define Backes (2006), tais características são denominadas assinaturas, que por sua vez são uma função ou matriz simplificada que representa e identifica o objeto. Sendo assim, estas assinaturas permitem a distinção entre objetos de classes diferentes. O mesmo vale para classificar e reconhecer imagens.

Na literatura há diversos métodos para análise de textura, tais como Local Binary Patterns (LBP ou Padrões Locais Binários), Gray-level Difference Method (GLDM ou Método de Diferenças de tons de cinza), Gray-level Cooccurrence Matrix (GLCM ou Matriz de Co-ocorrência), Descritores de Fourier e Gabor-Wavelets (FLORINDO, 2013).

Contudo, há uma abordagem mais recente baseada em descritores fractais em Casanova et al. (2016) e Marasca (2016), sendo esta última a alternativa utilizada neste trabalho, pois apresenta melhoria do algoritmo desenvolvido em Casanova et al. (2016). Além disso, esta abordagem tem mostrado bons resultados na análise de textura.

2.1.1 DESCRITORES FRACTAIS NA ANÁLISE DE TEXTURA

Durante tempos acreditou-se que a Geometria Euclidiana era capaz de explicar tudo que existia na natureza, pois se apoiava em formas perfeitas como retas, círculos, quadrados e cones. Contudo, Benoit Mandelbrot introduziu um novo conceito que mudou este panorama:

Geometria Fractal.

A Geometria Fractal é um ramo capaz de descrever e modelar os objetos naturais, considerando as imperfeições das formas, diferentemente da Geometria Euclidiana. Nos estudos de Mandelbrot ele mostrou que quando a parte de um desenho é ampliada, a parte menor se parece muito com a parte maior, apresentando auto-semelhança em diferente níveis de escala (BACKES, 2006; CASANOVA et al., 2016).

Conforme explica Casanova et al. (2016), os descritores fractais são utilizados para estimar a dimensão fractal de um objeto de interesse em diferentes níveis de escala. A dimensão fractal é uma medida que captura a complexidade do objeto, sendo a complexidade uma medida representativa dos detalhes encontrados em diferentes escalas. Assim, esta medida está relacionada com as características físicas do objeto, portanto, nos permite descrever e identificá-lo.

O método implementado por Casanova et al. (2016) e aprimorado em Marasca (2016), chamado de FD_{MIC}, faz uso do método de estimação da dimensão fractal de Bouligand-Minkowski para extrair as características de imagens, possibilitando a análise e reconhecimento de texturas.

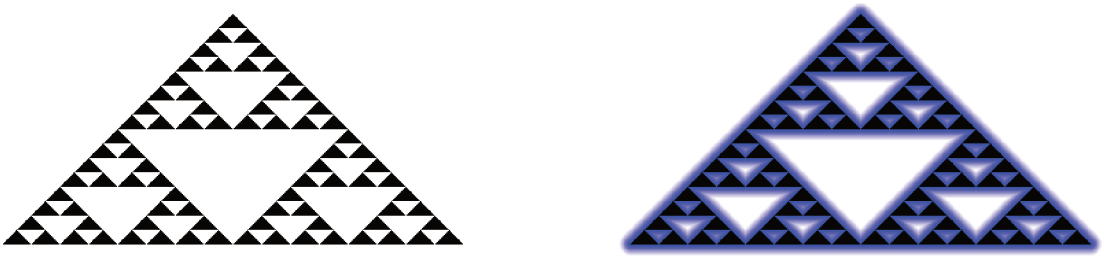
2.1.2 MÉTODO FDMIC

O método FD_{MIC}, como citado anteriormente, baseia-se na Geometria Fractal para extrair características de imagens, mais especificamente na dimensão fractal de Bouligand-Minkowski (BM) (BACKES, 2006).

Como exemplificação do cálculo da dimensão fractal de BM (DF_{BM}), realiza-se o cálculo sobre o Triângulo de Sierpinski, mostrado na Figura 1 (a). Inicialmente, cada ponto da imagem original é dilatado até um valor de raio máximo, simbolizado por r_{max}, como apresenta a Figura 1 (b), dilatada até r_{max} igual à sete (7).

Cada vez que um ponto da imagem é dilatado com determinado valor de raio (r), realiza-se o cálculo do seu volume. Dessa maneira, obtêm-se a relação $log(r) \times log(W(r))$ e traça uma reta $log(W) = \alpha \cdot log(R) + \beta$, na qual W é o vetor com valores de volume, R é o vetor com valores de raio e os coeficientes α e β são obtidos pelo método *Linear Least Squares* (em português Quadrados Mínimos Lineares ou Mínimos Quadrados) (GOLAN, 2012).

A dimensão fractal DF_{BM} é aproximada utilizando a Equação 1, na qual D é a dimensão na qual o objeto está sendo representado e (α) é o coeficiente angular da reta.



(a) Triângulo de Sierpinski (original)

(b) Triângulo de Sierpinski com dilatação até $r = 7$ pixels

Figura 1: Dimensão Fractal ideal, $DF = ln(3)/ln(2)$

Fonte: Marasca et al. (2018)

$$DF_{BM} = D - \alpha \tag{1}$$

Considerando o exemplo iniciado anteriormente, Marasca et al. (2018) apresenta na Figura 2 a relação entre raio e volume, bem como a reta gerada na dilatação do Triângulo de Sierpinski, com r_{max} igual à sete (7). Sendo assim, como o valor de α é igual à 0,41 e a dimensão a qual a imagem está representada é igual a dois (2), logo $DF_{BM} = 2 - 0,41 = 1,59$.

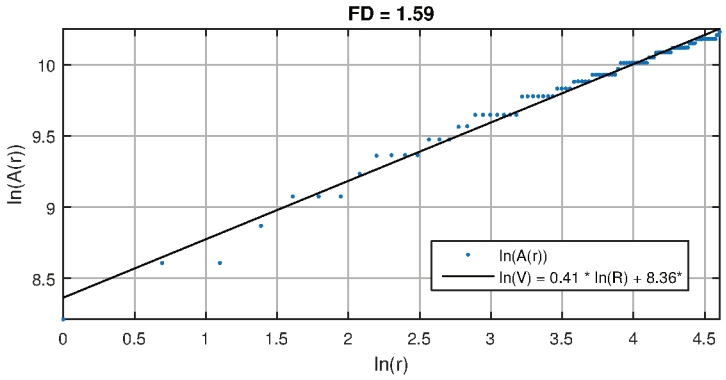


Figura 2: Representação da curva $\log(A(r)) \times \log(r)$ **e da regressão linear obtida após a dilatação do objeto da Figura 1.**

Fonte: Marasca et al. (2018)

Esta ideia de cálculo da dimensão fractal DF_{BM} foi utilizada como base para o desenvolvimento do método FD_{MIC}, que utiliza deste conceito para análise da textura. O método realiza um mapeamento de cada canal de uma imagem de textura no espaço de cores RGB para uma nuvem de pontos, que é considerada como um objeto fractal. Para este mapeamento, um *pixel* de posição *(i, j)* e intensidade *k*, é convertido em um *voxel* de coordenadas *(i, j, k)*, rotulado de acordo com seu canal de cor de origem (R, G ou B) (Figura 3

a).

A nuvem de pontos é dilatada como se todos os pontos da nuvem fossem um único objeto fractal, porém sempre preservando a informação do rótulo de cada *voxel* dilatado, assim como representado na Figura 3. A dilatação implementada pelo método realiza uma morfologia matemática utilizando o algoritmo de Transformada de Distância Euclidiana Exata proposto por Saito e Toriwaki (1994).

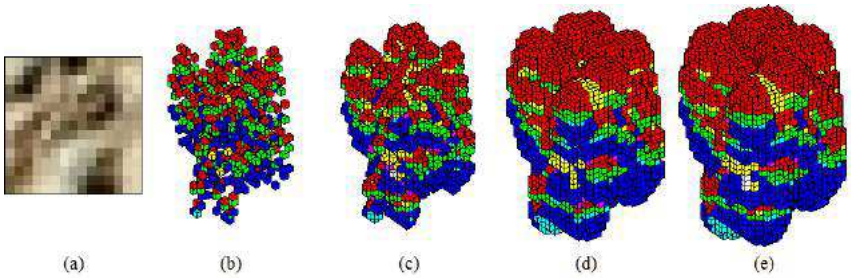


Figura 3: Exemplos da relação dos canais de cores no processo de dilatação. (a) Textura original; (b) nuvem de pontos original; nuvem de pontos dilatada para os raios (c) r = 1; (d) r = 3; (e) r = 4.

Fonte: Marasca (2016)

O cálculo da relação $ln(r) \times ln(V(r))$, no qual V é valor de volume, é realizado para cada rótulo, ou seja, para cada canal de cor, considerando agora que os pontos dilatados pertencentes a um determinado rótulo são objetos fractais. Logo, como existem três rótulos, considera-se existir três objetos fractais, sendo assim, realiza-se o processo de estimativa da dimensão fractal de BM para cada um deles. Contudo, o cálculo da dimensão fractal (Equação 1) não é realizado, considerando os próprios valores da relação $ln(r) \times ln(V(r))$ como características, resultando em três curvas de descrição fractal. Por fim, as três curvas, que são um conjunto de três vetores, são concatenadas em um único vetor, que possui uma linha e n colunas (esta medida depende do tamanho do vetor de cada rótulo), gerando o vetor de características fractais do método FD_{MIC}.

Imediatamente antes deste processo de extração de características, realiza-se o cálculo dos padrões de complexidade relacional entre os *pixels*. Este cálculo está relacionado com a quantização da imagem de entrada, sendo definida como a discretização dos níveis de intensidade luminosa de uma imagem, ou seja, no mundo real os níveis de intensidade luminosa são contínuos, mas no processo de digitalização de uma imagem os níveis de intensidade são discretizados, por padrão, em uma amplitude entre 0 e 255. Para redução da quantização de uma imagem pode-se utilizar a Equação 2, na qual I_{256} representa a imagem original com 256 níveis de intensidade luminosa e I_z é a imagem com z níveis de intensidade luminosa.

A etapa de quantização não pertence ao FD_{MIC}, no entanto é um processo intrínseco a ele, pois segundo Casanova et al. (2016) quanto menor o valor de z mais informações são perdidas durante a quantização, porém quanto maior o valor de z, menores informações são obtidas no cálculo da complexidade relacional entre os *pixels*. Por este motivo, foram realizados testes com valores 64, 96 e 256, afim de verificar qual resulta na melhor acurácia.

$$I_z = \frac{z-1}{255} I_{256} \tag{2}$$

2.2 APRENDIZAGEM DE MÁQUINA

O vetor de características, por si só, não é capaz de realizar a análise de textura de uma imagem e então classificá-la. Para tanto, é necessário a utilização de métodos de aprendizagem de máquina, ou seja, métodos que oferecem à uma máquina a capacidade de aprender por meio de dados e realizar predições, sendo uma alternativa mais eficiente do que os humanos (RASCHKA, 2015).

Conforme explica Raschka (2015), há 3 tipos de aprendizagem de máquina: supervisionada, não supervisionada e por reforço. A primeira, consiste em aprender a partir de dados já rotulados, dessa maneira sabe-se qual será o rótulo de saída baseado em dados de entrada. Por outro lado, na aprendizagem não supervisionada, há ausência dos rótulos, sendo assim o método de aprendizagem irá explorar e extrair informações significantes dos dados por conta própria. Por último, a aprendizagem por reforço, baseia-se em um agente que interage com o ambiente, a cada ação de interação existe uma recompensa que mede o quão boa foi aquela ação, o objetivo é maximizar esta recompensa a cada interação.

Vários métodos de aprendizagem de máquina supervisionada são apresentados na literatura. Neste presente trabalho, baseando-se nos resultados descritos na Seção 6.1.2 e considerando a vasta aplicação em problemas de classificação, optou-se pela utilização da análise discriminante (*Discriminant Analysis*), uma aprendizagem supervisionada, mais especificamente o classificador *Linear Discriminant Analysis* (LDA), introduzido por Fisher (1936).

2.2.1 LINEAR DISCRIMINANT ANALYSIS (LDA)

O LDA, como dito anteriormente, foi introduzido por R. Fischer em 1936. Segundo Cai et al. (2008) este é um método popular de extração de características que preserva a

separação das classes, ou seja, mantém classes distintas distantes uma das outras enquanto exige que pontos de uma mesma classe estejam próximos. Ele é mais indicado para situações em que os dados apresentem alta correlação, sendo o caso do vetor de características gerado pelo FD_{MIC}.

Supondo um tamanho G de populações numéricas, possuindo uma distribuição normal multivariada com uma matriz de covariância ($\sum$) comum, de dimensão pxp, e vetores de média μ_g $(g = 1, ..., G)$. Dada uma amostra x_i com rótulo de classe desconhecido, o objetivo é descobrir à qual classe g esta amostra pertence (GUO et al., 2007).

A ideia por trás do LDA é classificar esta amostra x_i para uma população $\tilde{g}$ de forma que minimize a Equação 3 . Isto significa encontrar a população que maximiza a probabilidade de observação (Equação 4) (GUO et al., 2007). Em outras palavras, o LDA busca encontrar qual das populações possui mais relação com a amostra x_i.

$$\tilde{g} = (x_i - \mu_g)^T \sum^{-1} (x_i - \mu_g) \tag{3}$$

$$\tilde{g} = argmin_g (x_i - \mu_g)^T \sum^{-1} (x_i - \mu_g) \tag{4}$$

2.3 VALIDAÇÃO ESTATÍSTICA

Afim de garantir a boa performance do método de aprendizagem e do classificador escolhido (LDA) sobre o vetor de características gerado pelo FD_{MIC}, torna-se necessário realizar uma avaliação estatística, pois conforme destaca Visa et al. (2011), uma boa acurácia na classificação é a preocupação primária. A performance está relacionada com a capacidade do método de prever sobre os dados de teste, indicando a sua qualidade e por consequência orienta a escolha do mesmo (HASTIE et al., 2009).

A performance também relaciona-se com a normalização dos dados. A aplicação de um método de normalização pode melhorar a precisão, a eficácia e, consequentemente, os resultados do algoritmo classificador. Existem diversos métodos para este fim e neste trabalho escolheu-se aplicar a distribuição normal padrão (*z-score*) (SHALABI et al., 2006).

Na prática, a performance é medida por meio do erro de predição. Contudo, no mundo real, este erro não pode ser calculado com exatidão, ele só pode ser estimado. Cada classificador possui um erro de predição associado, sendo esta uma variável aleatória medida, geralmente, por meio de seu viés e variância (RODRIGUEZ et al., 2010).

Deste modo, é importante escolher uma técnica de estimativa de erro adequada. Existem várias delas na literatura, mas para o presente trabalhou decidiu-se utilizar a validação cruzada *k-fold* ou *k-fold cross validation*, que segundo Hastie et al. (2009) e Rodriguez et al. (2010) é provavelmente a técnica mais simples e popular para este fim. Aliado à ela, há a matriz de confusão que auxilia na visualização dos erros e acertos do classificador.

2.3.1 DISTRIBUIÇÃO NORMAL PADRÃO

De acordo com Devore (2011) e Walpole et al. (2007) a distribuição normal é a mais importante na área de estatística e probabilidade, pois é capaz de descrever e explicar muitos fenômenos na natureza, na indústria e em pesquisas por meio da curva normal ou comumente conhecida como distribuição em forma de sino ou *bell-shaped*. Exemplos de aplicações estão em experimentos meteorológicos, medições de inteligência e aptidão, indicadores econômicos e experimentos científicos.

Uma distribuição normal é dita distribuição normal padrão quando $\mu = 0$ e $\sigma = 1$. Para alcançar tal objetivo é preciso reduzir um determinado valor x à um valor z, por meio da Equação 5, na qual z representa a normalização do valor de x, e μ e σ representam a média e o desvio padrão, respectivamente, de todas as instâncias.

$$z = \frac{x - \mu}{\sigma} \tag{5}$$

Dessa maneira a distribuição torna-se simétrica baseada na média e o z representará a distância do valor x em relação à média em unidades de desvio padrão. Assim, independente da forma como os dados se pareçam ou se existem valores atípicos (conhecidos pelo termo em inglês *outliers*), a normalização z-score é capaz de padronizá-los, objetivando igualar a influência dos mesmos no contexto que for aplicado.

Neste trabalho, a normalização é utilizada para normalizar cada índice do vetor de características gerado pelo FD_{MIC}. Para tanto, deve calcular para cada índice a média e o desvio padrão entre as instâncias.

2.3.2 VALIDAÇÃO CRUZADA K-FOLD

Basicamente, a validação cruzada *k-fold* divide todo o conjunto de dados em k subconjuntos de dados com tamanhos aproximadamente ou exatamente iguais. Com isso, o modelo classificador é treinado utilizando $k-1$ subconjuntos, enquanto que o subconjunto

restante (i) é utilizado para validar e testar o classificador. Este processo ocorre k vezes, sendo que a cada iteração o subconjunto de validação i é alterado, de forma que $i = 1, ..., k$.

A cada iteração calcula-se a taxa de erro de classificação. Ao término das repetições, a estimativa de erro é a média das taxas de erro cometidas em cada etapa. Sendo assim, segundo Rodriguez et al. (2010) a validação cruzada por *k-fold* depende de dois fatores: os subconjuntos de treinamento e a forma como são criados os k subconjuntos. Neste trabalho foi adotado a metodologia de k sendo igual a 10, sendo frequentemente utilizada e conhecida como validação *10-fold*.

A Figura 4 representa de forma gráfica a validação *10-fold*, na qual cada linha representa as iterações. Cada iteração contém retângulos que representam os subconjuntos, nos quais o retângulo colorido é o subconjunto utilizado para teste do classificador e o restante para treiná-lo.

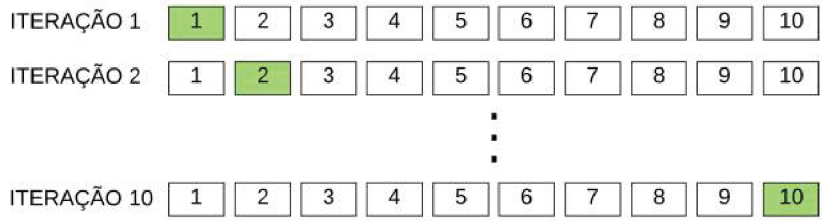


Figura 4: Representação da validação cruzada *10-fold*

Fonte: Autoria Própria

2.3.3 MATRIZ DE CONFUSÃO

A matriz de confusão é uma técnica efetiva para avaliação de um classificador, já que é capaz de indicar a sua taxa de acerto e de erro. Consideremos uma matriz 2 x 2 para classificação de dados nas classes *Sim* e *Não*, conforme Quadro 1. O *verdadeiro positivo* (VP) e o *verdadeiro negativo* (VN), representam as amostras classificadas corretamente. O *falso positivo* (FP), representa amostras quem foram classificadas como *Sim*, mas na realidade eram *Não* e o *falso negativo* (FN), representa o contrário disso, amostras classificada como *Não*, mas na realidade eram *Sim*.

Segundo Visa et al. (2011), podemos calcular a taxa de acerto ou acurácia (Equação 6) e de erro (Equação 7) do classificador da seguinte maneira:

$$Acurcia = \frac{VP + VN}{VP + FP + VN + FN} \tag{6}$$

		Classe classificada	
		Sim	**Não**
Classe atual	**Sim**	Verdadeiro Positivo	Falso Negativo
	Não	Falso Positivo	Verdadeiro Negativo

Quadro 1: Matriz de confusão para um problema de duas classes

Fonte: Witten et al. (2016)

$$Erro = \frac{FP + FN}{VP + FP + VN + FN} \tag{7}$$

Conforme explica Witten et al. (2016), em classificações de múltiplas classes a matriz de confusão terá duas dimensões (linhas e colunas) e cada classe terá sua linha e coluna, representando a classe atual e a classe classificada, respectivamente. Bons resultados serão representados por grandes valores na diagonal e pequenos valores fora da diagonal, sendo 0 (zero) o ideal.

3 APLICAÇÕES DE ANÁLISE DE TEXTURA

De modo geral, já existem aplicações no mercado em diversas áreas que utilizam da análise de textura, contudo há uma concentração de trabalhos publicados na medicina, na área industrial, na agricultura e na botânica.

As duas primeiras, apresentam aplicações mais robustas, pois exigem um elevado processamento computacional para gerar resultados em tempo real e com alta confiabilidade. Por outro lado, as outras duas, que estão intimamente relacionadas, apresentam aplicações simplificadas com objetivo, principalmente, de melhorar e agilizar atividades humanas corriqueiras. Esta seção busca apresentar de forma concisa alguns estudos em cada área, bem como as aplicações.

3.1 ÁREA MÉDICA

Dentre algumas abordagens na medicina, há diversos estudos à respeito de análise de texturas aplicadas à diagnósticos médicos por imagens, como: detecção de doenças de pele, câncer de mama, glaucoma e doenças no fígado.

Uma das aplicações neste sentido, é o primeiro software para análise quantitativa da textura de imagens chamado MaZda (SZCZYPIŃSKI; KLEPACZKO, 2017; SZCZYPIŃSKI et al., 2009). Seu desenvolvimento é datado de 1996 e inicialmente foi idealizado para análise de imagens de ressonância magnética, porém mostrou-se similarmente efetivo para raio-x e imagens capturadas por câmeras. É uma ferramenta livre e de código-aberto escrita na linguagem C++ e Delphi utilizando a biblioteca de computação gráfica OpenGL. Está disponível para os sistemas operacionais Windows, Linux e OS X.

O software permite análise de imagens 2D e 3D, neste último apenas imagens específicas como ressonância magnética e tomografia computadorizada. O usuário pode definir regiões de interesses na imagem, chamadas de *region of interest* (ROI), concentrando o esforço computacional e evitando processamento desnecessário.

Em relação à textura o software considera três situações: caracterização, segmentação e classificação. A primeira descreve de forma numérica a textura da imagem, a segunda particiona a imagem em regiões, cada uma representa um conjunto de textura e a última determina à qual classe de textura pertence cada região.

Basicamente o processo de análise divide-se na entrada de imagens (2D ou 3D), mapeamento e seleção de características e por fim a classificação e visualização do resultado.

3.2 ÁREA INDUSTRIAL

Na indústria moderna, uma das principais preocupações está relacionada à inspeção de defeitos e garantia de qualidade na linha de produção. Como exemplo, os trabalhos de Kumar (2008), Mahajan et al. (2009) e Ngan et al. (2011) realizam uma extensa revisão à respeito de técnicas automatizadas para detecção de defeitos na indústria têxtil. Neste contexto, a análise de textura é a base para extrair características da textura dos produtos, analisá-las e então encontrar defeitos, caso existam.

No mesmo segmento, as aplicações tradicionalmente são robusta já que estão inseridas em um ambiente que necessita de respostas em tempo real. Além disso alinham um alto poder de processamento entre hardware e software, como é o caso do BMSVision Cyclops mostrado na Figura 5 (GOYAL, 2018).

Figura 5: Sistema BMSVision Cyclops

Fonte: BMSVision (2018)

O funcionamento do sistema consiste de uma câmera que, auxiliada por um sistema de iluminação, digitaliza imagens do tecido, as transfere para a unidade de processamento de imagem que utiliza de algoritmos para analisar a textura do tecido e detectar desvio do padrão.

Algumas das vantagens da máquina são: detecção em tempo real, caso um tecido seja considerado defeituoso a produção é interrompida, ele não necessita da percepção humana e possui integração com sistemas de monitoramento. Além disso, possui um sistema inteligente para realizar calibrações necessárias de forma automática como câmera, iluminação e largura da digitalização.

3.3 ÁREA AGRÁRIA E BOTÂNICA

Como dito anteriormente a agricultura e a botânica estão relacionadas, portanto podem ser consideradas como um grupo único. Há diversos estudos nestas áreas relacionados principalmente às plantas. O trabalho de Chaudhari et al. (2016) realiza uma pesquisa à respeito de trabalhos sobre detecção de plantas doentias utilizando processamento de imagem. Wäldchen e Mäder (2018) realiza um mapeamento sistemático sobre técnicas de visão computacional para a identificação da espécie de plantas e Wäldchen et al. (2018) além de revisar o estado da arte no ramo, expõe desafios e tendências.

Exemplo de aplicação que utiliza a análise de textura é o aplicativo de recuperação e compartilhamento de imagem para identificação de plantas, chamado Pl@ntnet (JOLY et al., 2014) e (GOëAU et al., 2014). Atualmente está disponível para Android, iOS e versão web, além disso abrange espécies da flora da Europa, da América do Sul e do norte da África.

O funcionamento baseia-se em o usuário enviar fotos da mesma planta, assinalando qual parte a foto representa (flor, folha, fruta ou casca). O sistema por sua vez irá extrair características locais, como forma e textura, para classificar a espécie e encontrar aproximações. Os registros encontrados são retornados ao usuário com uma pontuação de confiança em ordem decrescente, bem como as respectivas imagens. Caso ele encontre a espécie que procura ele poderá compartilhar esta observação, que consequentemente aparecerá em uma outra ferramenta que permite à comunidade revisar e avaliar para então incluí-la no sistema.

4 MATERIAIS E METODOLOGIA

Neste capítulo estão descritos os materiais e a metodologia utilizados no decorrer deste trabalho com intuito de alcançar os objetivos propostos e descritos na Seção 1.2, tendo como objetivo principal o desenvolvimento de uma aplicação móvel capaz de classificar uma imagem por meio da análise de textura baseada na teoria fractal.

4.1 MATERIAIS

Esta seção descreve os materiais utilizados para o desenvolvimento da aplicação proposta neste trabalho. Foram necessários ambientes de desenvolvimento integrado (IDE's) para as plataformas *desktop* e móvel, bibliotecas específicas para o processamento de imagem e aprendizagem de máquina e, por fim, a utilização de bases de imagens com intuito de validar o método de análise implementado.

4.1.1 AMBIENTES DE DESENVOLVIMENTO INTEGRADO - IDE'S

As IDE's escolhidas para desenvolver aplicações para plataformas *desktop* e móvel foram, respectivamente, NetBeans e Android Studio, sendo ambas disponibilizadas de forma gratuita.

A IDE Netbeans possui suporte à diversas linguagens, entre elas a linguagem Java, além de ser compatível com o sistema operacional Windows. Possui uma ferramenta integrada para a criação de interfaces gráficas de forma ágil e simples baseando-se na biblioteca de componentes do Java Swing, chamada de *GUI Builder*. Além disso possibilita a integração com bibliotecas de aprendizagem de máquina e visão computacional.

O Android Studio é uma IDE, compatível com Windows, destinada ao desenvolvimento de aplicativos móveis para o sistema Android. Apresenta funcionalidades e ferramentas que agilizam o processo de desenvolvimento, permitindo desenvolver para todos os aplicativos Android em um único ambiente. Como a IDE anterior, também permite a integração

com bibliotecas de aprendizagem de máquina e visão computacional.

4.1.2 BIBLIOTECAS

A aplicação utilizará técnicas de processamento de imagens e aprendizagem de máquina empregando duas bibliotecas de código aberto que disponibilizam funções para estes fins.

A biblioteca de processamento de imagem escolhida foi o OpenCV. Ela é compatível com diversas plataformas, incluindo Windows e Android. Possui mais de 2.500 algoritmos otimizados para algumas linguagens, incluindo Java. Foi elaborada para aplicações em tempo real. Permite, entre diversas funcionalidades, manipulação e transformações de imagens de forma simplificada.

A biblioteca de aprendizagem de máquina escolhida foi o Weka, que contém algoritmos de aprendizagem de máquina para tarefas de mineração de dados. Ela pode ser integrada à um software desenvolvido em Java, bem como em aplicações Android, agregando todas as suas funcionalidades como pré-processamento de dados, classificação, regressão e visualização dos resultados.

4.1.3 BASES DE IMAGENS

A fim de verificar e validar a acurácia do método de análise FD_{MIC} implementado na linguagem Java, utilizou-se duas bases de imagens como *benchmark*: Outex e Vistex.

A base Outex (Figura 6) consiste em um *benchmark* para avaliação empírica de algoritmos de análise de textura. Contém uma vasta coleção de texturas de superfícies e cenas naturais capturadas em diferentes condições. Neste trabalhou utilizou-se a base de dados para testes com ID Outex_TC_00013, contendo 68 texturas, com 20 imagens para cada textura (OJALA et al., 2002).

A Vistex (Figura 7) também é uma coleção de imagens de textura. Ela fornece um grande conjunto de texturas de alta qualidade para aplicações de visão computacional de forma gratuita. Além disso disponibiliza imagens que representam condições do mundo real, ou seja, não seguem padrões de iluminação e angulação, incluindo muitas texturas não tradicionais (MIT, 1995).

Figura 6: Amostras de imagens contidas na base Outex

Fonte: Autoria Própria

Figura 7: Amostras de imagens contidas na base Vistex

Fonte: Autoria Própria

4.2 METODOLOGIA

A metodologia definida para este trabalho divide-se em três etapas. Primeiro implementou-se o FD_{MIC} na linguagem Java e realizou-se testes de validação. No segundo momento, desenvolveu-se o software *desktop* para geração do modelo classificador. Por último, desenvolveu-se a aplicação móvel para receber o modelo e posteriormente classificar imagens. As três etapas estão descritas na sequência.

4.2.1 IMPLEMENTAÇÃO E MEDIÇÃO DO FDMIC

Como mencionado anteriormente, o método de extração de características escolhido foi o FD_{MIC} aprimorado em Marasca (2016). Contudo, tornou-se necessário implementá-lo na linguagem Java para torná-lo compatível com a aplicação móvel. Em sequência à esta etapa, desenvolveu-se um software, também em Java, com a finalidade de validar os resultados do método implementado. Ademais, partes do código deste software foram reutilizadas para o desenvolvimento da aplicação.

Para desenvolver este software de validação foram realizados os passos listados a seguir, que estão resumidos no Pseudocódigo 1:

1. Criar uma estrutura para carregar imagens uma a uma, a partir de uma base de imagens;
2. Criar uma classe contendo o método FD_{MIC}, para gerar o vetor de características de cada imagem passada como parâmetro;
3. Concatenar cada vetor de características retornado pelo método para criar um *dataset*;
4. Carregar um arquivo com as classes de cada imagem de entrada, para gerar uma lista de classes;
5. Criar um arquivo compatível com o Weka, no formato ARFF (extensão *.arff*), utilizando o *dataset* e a lista de classes;
6. Carregar este arquivo ARFF e aplicar suas instâncias à um processo de validação cruzada e por fim obter as avaliações estatísticas.

A estrutura para carregar as imagens de entrada foi criada utilizando a biblioteca OpenCV. Ela é capaz de manipular imagens em diversos formatos, tais como *png* e *bmp* utilizados na validação. No entanto, durante a execução, as imagens são tratadas como uma

Algoritmo 1 Passo a passo do software desenvolvido para validação

Entrada: Imagens I
Saída: Avaliações estatísticas AE
1: **para** $i \leftarrow I_1 \ldots In$ **faça**
2: VCaracteristicas ← FDMIC(I_i);
3: Dataset$[i][:]$ ← VCaracteristicas;
4: **fim para**
5: ListaClasses ← Carrega lista de classes;
6: Gera arquivo ARFF(Dataset, ListaClasses);
7: Instâncias ← Carrega arquivo ARFF;
8: AE ← Validação Cruzada(Instâncias);

classe Mat, permitindo assim a utilização de funções para manipulação das imagens de forma simplificada.

O método de extração de características (FD_{MIC}) foi implementado como uma classe, na qual seu construtor padrão realiza operações custosas que necessitam ser calculadas uma única vez durante a execução. Esta classe possui um método que recebe uma única imagem e realiza o processo de extração de características, gerando o vetor de características referente a esta imagem. O método dispõe de uma constante denominada r_{max} que possui valor igual à cinco (5), definida com base em Marasca (2016). Com este valor, o FD_{MIC} gera o vetor de características com tamanho 66. Cada vetor gerado é concatenado em uma matriz, produzindo um *dataset* com as características de todas as imagens, este procedimento está representado entre as linhas 1 e 4 do Pseudocódigo 1.

Na linha 5 do Pseudocódigo 1, uma função é executada para gerar uma lista contendo os valores de classificação para cada uma das imagens utilizadas. Neste procedimento, realiza-se a leitura de um arquivo de texto, no qual cada linha representa a classe de uma imagem. Os valores então são convertidos para uma lista em Java. Utilizando esta lista e o *dataset* de características, é possível criar um arquivo compatível com o Weka por meio da função representada pela linha 6.

O Weka é a biblioteca responsável pela classificação das instâncias e medição dos resultados, contudo ela trabalha apenas com arquivos no formato ARFF (*Attribute – RelationFileFormat*). A Figura 8 apresenta a estrutura de um arquivo neste formato, que é a representação de um *dataset*.

```
1  @RELATION dataset
2
3  @ATTRIBUTE atributo0 real
4  @ATTRIBUTE atributo1 real
5  @ATTRIBUTE atributo2 real
6  @ATTRIBUTE classes {A,B,C}
7
8  @DATA
9  283454.0,285878.0,289568.0,A
10 182060.0,215237.0,234228.0,A
11 87699.0,150562.0,170766.0,B
12 242672.0,259941.0,271806.0,B
13 280924.0,283996.0,291826.0,C
14
```

Figura 8: Exemplo da estrutura de um arquivo no formato ARFF

Fonte: Autoria Própria

A declaração *@RELATION* define basicamente um título para a relação das instâncias. A declaração *@ATTRIBUTE* define o nome e tipo (inteiro, real, *string*, etc.) de cada atributo da instância. Neste trabalho há 66 atributos, pois o vetor de características possui esse tamanho. O último atributo, tradicionalmente, é reservado para definir quais são as classes englobadas pelo *dataset*, definindo-as uma a uma, separadas por vírgulas e entre chaves, conforme mostra a linha 6 da Figura 8. Por último, está a declaração *@DATA*, em que cada linha representará uma instância, contendo os atributos separados por vírgulas e o último atributo indicando à qual classe ela pertence.

Após carregar o arquivo ARFF, afim de obter as instâncias, os valores dos atributos são aplicados à uma função de normalização *z-score*, para então serem utilizados em uma outra função que realiza a validação cruzada por *10-fold* com o classificador LDA, gerando as avaliações estatísticas como matriz de confusão, taxas de acerto e de erro. Ambas as funções, de normalização e validação cruzada, são nativas da biblioteca.

4.2.2 DESENVOLVIMENTO DA APLICAÇÃO

A aplicação proposta neste trabalho está dividida em duas partes: geração do modelo e a classificação de imagens. A primeira é um software *desktop* e a segunda um aplicativo móvel.

4.2.2.1 SOFTWARE DESKTOP

O software *desktop* foi desenvolvido na linguagem Java e possui uma interface gráfica criada com a *GUI Builder* do NetBeans. Ele é responsável por receber um conjunto de imagens que, obrigatoriamente, devem estar organizadas em subpastas dentro de uma pasta raiz, cada uma dessas subpastas deverá ser nomeada de acordo com a classe referida e receberá somente imagens pertencentes à ela, conforme exemplifica a Figura 9. Além das imagens, é preciso informar alguns valores como parâmetros de entrada, tais como:

- Valor da escala: este valor indica à qual tamanho deseja-se reescalar as imagens, baseando-se na maior dimensão (altura ou largura). Com ele calcula-se o fator de reescala.
- Tamanho da subimagem: este valor define o tamanho (altura e largura) das subimagens que são recortadas das imagens de entrada.
- Altura e largura da imagem original: estes dois valores permitem, juntamente com o valor da escala, calcular o fator de reescala.

Destes valores, os dois primeiros são gravados em um arquivo de texto, denominado como configuração, que deve ser transferido para a aplicação móvel.

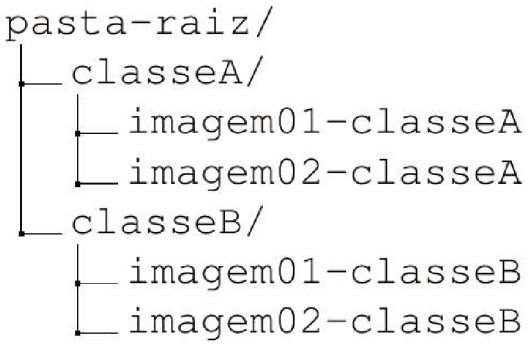


Figura 9: Exemplo da organização de imagens em pastas e subpastas
Fonte: Autoria Própria

Utilizando a base de imagens o software gera uma base de treinamento própria, ou seja, cada imagem é reescalada utilizando o fator de reescala e recortada em subimagens, seguindo o parâmetro de tamanho informado. Cada subimagem será nomeada de acordo com sua classe, gerando assim um *dataset* rotulado. A partir deste *dataset* será criado um arquivo ARFF com as instâncias que, subsequentemente, serão aplicadas à uma função do Weka responsável por

construir um modelo classificador treinado utilizando o classificador LDA. Ambos arquivos, modelo treinado e arquivo ARFF, também devem ser transferidos para a aplicação móvel. Todos os procedimentos descritos estão representados na Figura 10.

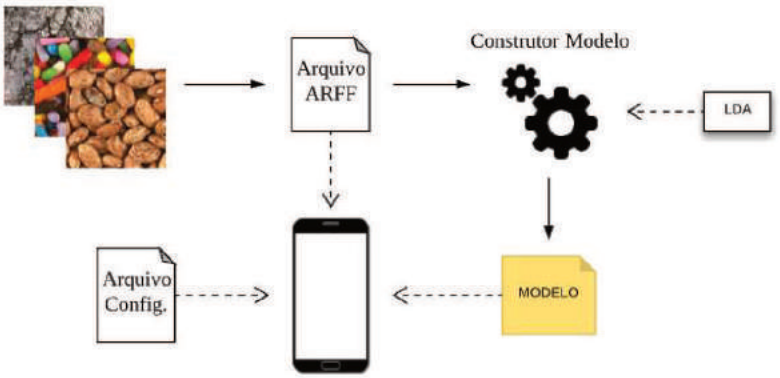


Figura 10: Fluxograma ilustrando os procedimentos para construção do modelo classificador
Fonte: Autoria Própria

4.2.2.2 APLICAÇÃO MÓVEL

O aplicativo móvel foi desenvolvido para o sistema Android. Ele recebe, de forma manual, o modelo treinado, o arquivo de configuração e o arquivo ARFF via conexão por cabo USB, sendo armazenado na pasta pública do aplicativo.

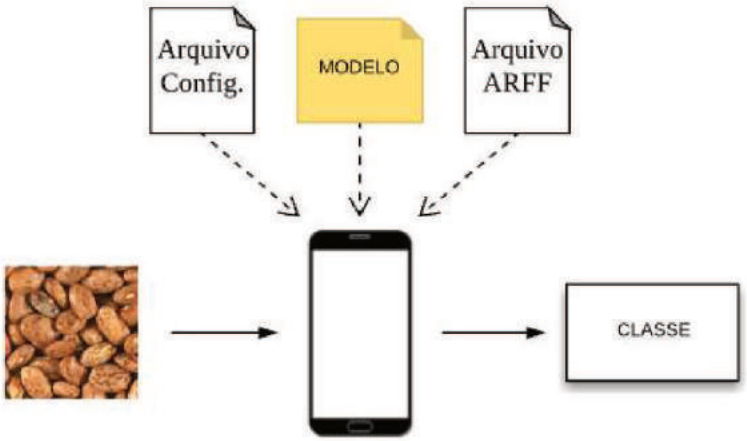


Figura 11: Fluxograma ilustrando o funcionamento da aplicação móvel
Fonte: Autoria Própria

Para classificar uma imagem, o usuário deve selecionar uma a partir da galeria ou capturá-la com a câmera, respeitando alguns requisitos com o propósito de obter uma imagem válida para a classificação. Ao selecionar, a imagem é reescalada no intuito de normalizá-la, pois ela pode conter uma resolução diferente das imagens utilizadas na geração do modelo. A aplicação permite que o usuário recorte uma subimagem que é utilizada para a classificação, respeitando o tamanho de subimagem constado no arquivo de configuração. Este funcionamento pode ser visualizado na Figura 11.

5 INTERFACE DAS APLICAÇÕES

Neste capítulo são apresentados os resultados das interfaces gráficas obtidas após o desenvolvimento das aplicações *desktop* e móvel.

5.1 INTERFACE GRÁFICA DO SOFTWARE DESKTOP

Como mencionado, a aplicação *desktop* foi desenvolvida utilizando a IDE NetBeans e a interface gráfica utilizando a ferramenta GUI Builder, nativa do NetBeans. A Figura 12 apresenta a versão final da aplicação.

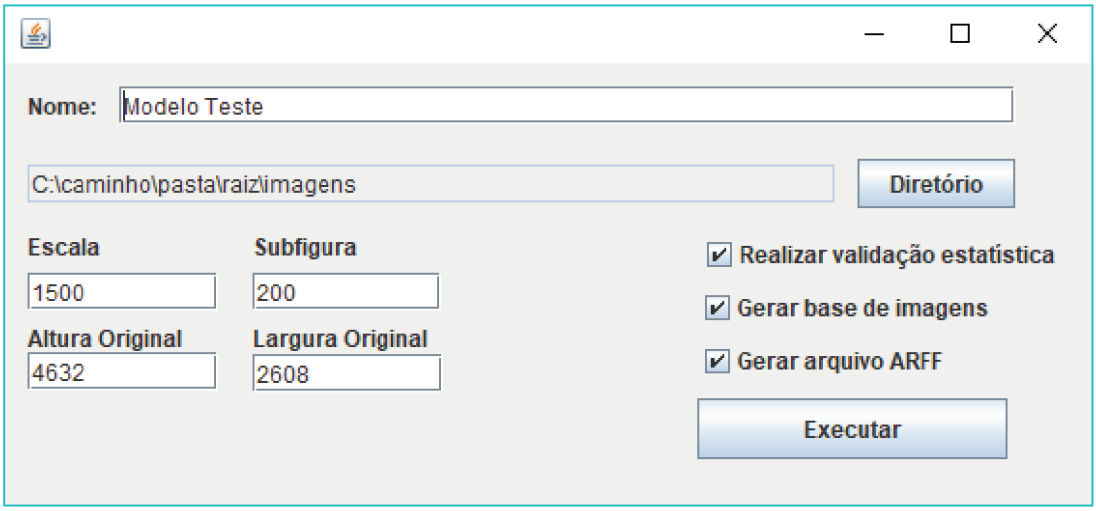


Figura 12: Interface gráfica do software *desktop*.

Fonte: Autoria Própria

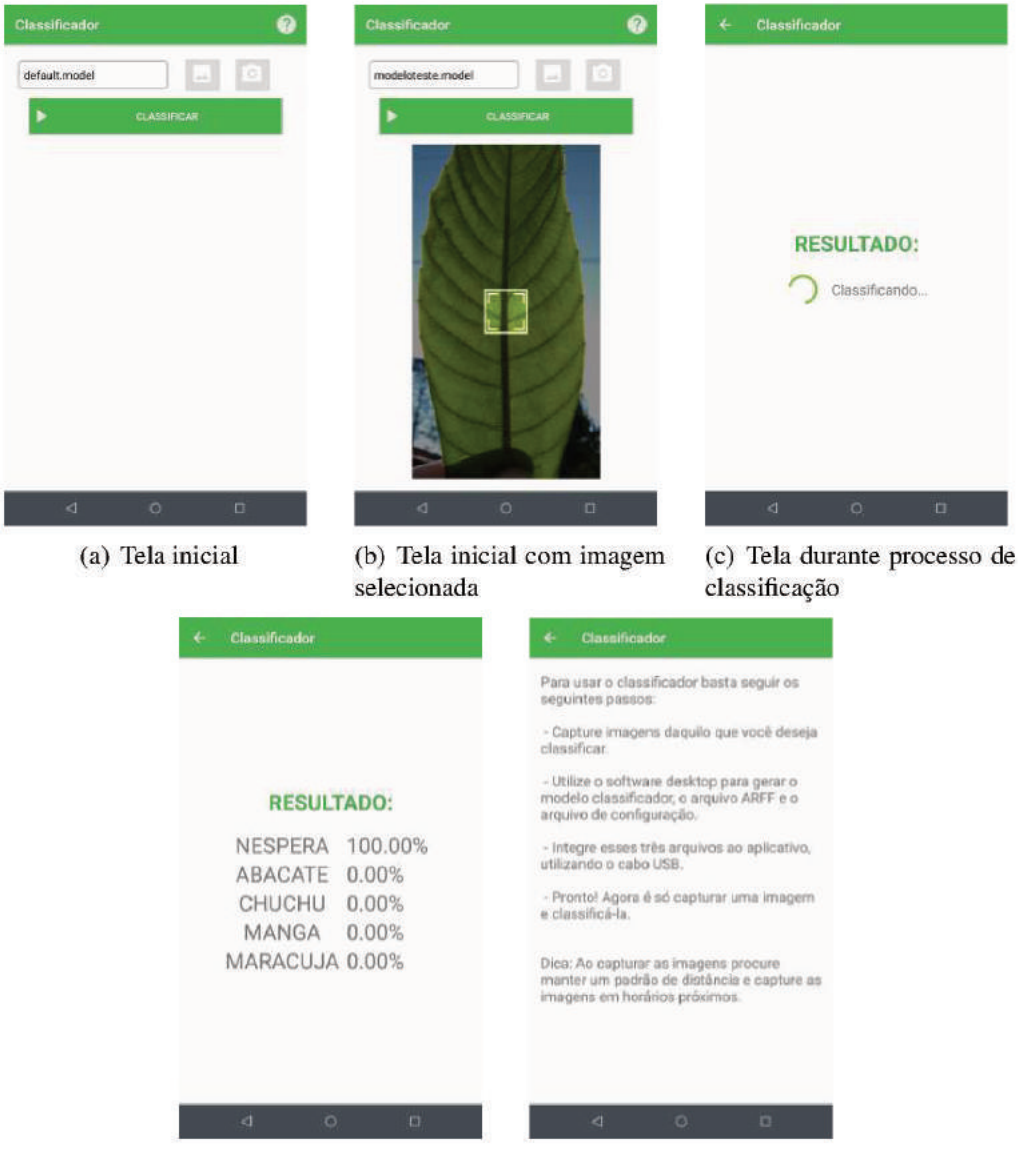
A interface possui um campo de texto para nomear o modelo a ser criado. Logo abaixo, um botão destinado à escolha do diretório onde estão as imagens que servirão para gerar a base de imagens e treinar o modelo. O caminho a ser escolhido deve apontar para a "pasta-raiz", conforme mostrado na Figura 9 da Seção 4.2.2.1.

Abaixo existem quatro campos de texto destinados à informar o valor da escala, tamanho da subimagem, a altura e a largura das imagens utilizadas para gerar o modelo, sendo

todos medidos em *pixels*. Há também três *checkboxes* que permitem ao usuário decidir se deseja realizar a validação cruzada, se deseja gerar a base de imagens para criar o modelo e se deseja gerar o arquivo ARFF. Por fim, o botão de executar para iniciar o processamento.

5.2 INTERFACE GRÁFICA DO APLICATIVO MÓVEL

A aplicação móvel, desenvolvida para Android, possui três telas definidas: Tela inicial (Figuras 13 (a) e 13 (b)), tela dos resultados (Figuras 13 (c) e 13 (d)) e tela de tutorial (Figura 13 (e)).



(a) Tela inicial

(b) Tela inicial com imagem selecionada

(c) Tela durante processo de classificação

(d) Tela com o resultado final

(e) Tela de tutorial

Figura 13: Telas do aplicativo móvel de análise de textura

Fonte: Autoria Própria

A tela inicial, mostrada na Figura 13 (a) apresenta um *combobox* com as opções de modelos classificadores integrados com o aplicativo. Ao lado direito há duas opções: escolher uma imagem da galeria ou capturar uma foto com a câmera. Logo abaixo o botão de classificar que inicia o processamento. Ao selecionar uma imagem, ela é apresentada na tela conforme mostra a Figura 13 (b). Neste estado, nota-se o quadrado de recorte da subimagem, que possui dimensões de acordo com o modelo que se esta utilizando.

Ao iniciar a classificação, a aplicação apresenta a tela da Figura 13 (c) enquanto extrai as características da imagem e a classifica. Quando este processamento finaliza, os resultados são apresentados conforme mostra a Figura 13 (d), na qual cada linha apresenta a denominação da classe e a probabilidade da imagem pertencer a esta classe. Por fim, existe a tela específica com instruções de como gerar e integrar o modelo classificador à aplicação, apresentada na Figura 13 (e), esta tela pode ser acessada clicando no ícone localizado no canto direito superior da barra de ações da tela inicial.

6 RESULTADOS

Neste capítulo são apresentados os testes de acurácia do método FD_{MIC}, implementado como uma classe Java, em relação as bases de *benchmarks*, nos quais são avaliados valores do parâmetro de quantização z e o uso de diferentes classificadores. Na sequência, é exposto o detalhamento sobre o estudo de caso, que inclui a descrição da captura das imagens para treinamento e para teste, as avaliações do parâmetro de quantização z, por meio da validação cruzada, sobre a base de treinamento e resultados obtidos ao avaliar o modelo gerado com a base de teste. Por fim, apresenta-se os resultados de testes de acurácia em abordagens alternativas: isolando variáveis (i.e. angulação, distância e iluminação), validação utilizando bases referentes ao mesmo *smartphone* e, repetindo alguns experimentos para imagens em tons de cinza.

6.1 BENCHMARKS

Nesta seção estão descritos os resultados obtidos por meio do software descrito na Seção 4.2.1, que mediu a acurácia do método FD_{MIC} em relação aos *benchmarks* Outex e Vistex, apresentados na Seção 4.1.3. Em seguida, realizou-se avaliações do método utilizando diferentes classificadores.

6.1.1 TESTE QUANTIZAÇÃO

Para cada base, Outex e Vistex, realizou-se três (3) testes utilizando o classificador LDA, nos quais foram alterados os valores de quantização z entre 64, 96 e 256, com o intuito de descobrir qual valor extrai informações mais ricas a respeito da textura, melhorando a acurácia do método.

A Tabela 1 apresenta informações gerais à respeito das bases de imagens utilizadas. As Tabelas 2 e 3 apresentam os resultados dos testes executados, respectivamente, na base Outex e na Vistex, obtidos por meio da validação cruzada *10-fold*.

Tabela 1: Informações gerais sobre as bases de imagens utilizadas

Informações	Base Outex	Base Vistex
Número de Imagens	1360	864
Qtd de Classes	68	54

Fonte: Autoria própria.

Tabela 2: Resultados obtidos nos testes de quantização z na base Outex

Valor z	Número Instâncias	Instâncias Corretas	Instâncias Erradas	Taxa de Acerto (%)	Taxa de Erro (%)
64	1360	1248	112	91,7647	8,2353
96	1360	1258	102	92,50	7,50
256	**1360**	**1269**	**91**	**93,3088**	**6,6912**

Fonte: Autoria própria.

Tabela 3: Resultados obtidos nos testes de quantização z na base Vistex

Valor z	Número Instâncias	Instâncias Corretas	Instâncias Erradas	Taxa de Acerto (%)	Taxa de Erro (%)
64	864	855	9	98,9583	1,0417
96	864	854	10	98,8426	1,1574
256	**864**	**857**	**7**	**99,1898**	**0,8102**

Fonte: Autoria própria.

Conforme os resultados, para estas bases de imagens, as melhores taxas de acerto ocorreram ao utilizar o valor z igual à 256. Contudo, é possível perceber que a taxa de acerto com outros valores de z oscilam minimamente em relação à maior taxa obtida. Além disso, nota-se que o valor de z não segue um padrão nos resultados, ou seja, não significa que dependendo do valor z o resultado será melhor ou pior, pois o resultado depende de qual contexto o método está sendo aplicado. Por fim, de modo geral, o método apresentou bons resultados com as bases de *benchmarks*, pelo fato de que todas as taxas de acerto foram acima de 90%.

6.1.2 TESTE CLASSIFICADORES

Afim de avaliar outros classificadores apresentados na literatura e compará-los com o LDA, utilizou-se as mesmas bases de *benchmark*, porém manteve-se o valor da quantização z fixo em 256, pois apresentou os melhores resultados conforme descrito na Seção 6.1.1. Os classificadores testados foram: k-Nearest Neighbors (k-NN), Naive Bayes, Support Vector Machines (SVM), Random Forests e Multilayer Perceptron (RASCHKA, 2015; SHALEV-SHWARTZ; BEN-DAVID, 2014). Os resultados sobre as bases Outex e Vistex foram obtidos

por meio da validação cruzada *10-fold* e são apresentados, respectivamente, nas Tabelas 4 e 5, juntamente com o valor obtido pelo LDA nos testes anteriores.

Tabela 4: Resultados obtidos nos testes de classificadores na base Outex

Classificadores	Número Instâncias	Instâncias Corretas	Instâncias Erradas	Taxa de Acerto (%)	Taxa de Erro (%)
k-NN	1360	1142	218	83,9706	16,0294
Naive Bayes	1360	760	600	55,8824	44,1176
SVM	1360	889	471	65,3676	34,6324
Random Forests	1360	1085	275	79,7794	20,2206
Multilayer Perceptron	1360	1186	174	87,2059	12,7941
LDA	**1360**	**1269**	**91**	**93,3088**	**6,6912**

Fonte: Autoria própria.

Tabela 5: Resultados obtidos nos testes de classificadores na base Vistex

Classificadores	Número Instâncias	Instâncias Corretas	Instâncias Erradas	Taxa de Acerto (%)	Taxa de Erro (%)
k-NN	864	730	134	84,4907	15,5093
Naive Bayes	864	552	312	63,8889	36,1111
SVM	864	594	270	68,75	31,25
Random Forests	864	686	178	79,3981	20,6019
Multilayer Perceptron	864	793	71	91,7824	8,2176
LDA	**864**	**857**	**7**	**99,1898**	**0,8102**

Fonte: Autoria própria.

A partir dos resultados, nota-se que para a análise de textura utilizando o método FD_{MIC}, a melhor abordagem é a utilização do classificador LDA, o qual apresentou os melhores resultados em ambos os testes.

6.2 ESTUDO DE CASO

Nesta seção detalha-se os testes realizados sobre o estudo de caso, bem como a construção das bases de imagens e os resultados obtidos. O estudo permitiu avaliar a acurácia do aplicativo de análise de textura, baseado na teoria fractal, em aplicações nas quais variáveis como rotação, escala, cor e iluminação não são precisamente controladas. O estudo se tratou da classificação de espécies de plantas nativas situadas na UTFPR, câmpus Dois Vizinhos - PR.

6.2.1 CAPTURA DAS IMAGENS

As imagens utilizadas para gerar a base de treinamento e de teste foram coletadas no Viveiro de Mudas Hortícolas, situado na fazenda da UTFPR, do câmpus Dois Vizinhos -

PR. Ao todo foram capturadas cinco (5) imagens de cada uma das sete (7) espécies de plantas escolhidas: Abacate, Araça Amarelo, Araça Vermelho, Chuchu, Manga, Maracujá e Nêspera. A partir destas 35 imagens foram cortadas subimagens que formaram as bases, conforme detalhado na Seção 6.2.2.

Como o estudo tinha por objetivo testar o modelo classificador treinado com imagens de um determinado *smartphone* em diferentes resoluções de câmera, as imagens foram capturadas por dois dispositivos distintos, especificados no Quadro 2 juntamente com as dimensões das imagens capturadas por ambos.

Dispositivo	**Dimensão (A x L)**	**Base**
Motorola Moto G5S	4632 x 2608	Treinamento
Asus Zenfone 5	3264 x 2448	Teste

Quadro 2: Informações dos modelos de *smartphones* utilizados na captura de imagens do estudo de caso

Fonte: Autoria própria.

Todas as imagens foram capturadas no mesmo dia e no mesmo período de tempo, entre 14h e 16:30h, sendo assim nas mesmas condições de iluminação, neste caso, solar. Além disso, respeitou-se as seguintes instruções:

- Folha da planta posicionada contra o sol;
- Câmera posicionada há uma distância entre 10 e 15 centímetros da folha;
- Superfície da folha e da câmera paralelas uma a outra, ou seja, sem qualquer ângulo de inclinação, no momento da captura.

6.2.2 TESTE QUANTIZAÇÃO

Semelhantemente ao teste descrito na Seção 6.1.1, o mesmo foi realizado sobre as imagens de treinamento, ou seja, foram alterados os valores da quantização z entre 64, 96 e 256. Para tanto, utilizou-se o software apresentado nas Seções 4.2.2.1 e 5.1, dessa maneira também foram realizados testes de escalas (variando entre 1000 e 2500) e o tamanho das subimagens (variando entre 100 e 400). Os resultados, obtidos por meio da validação cruzada *10-fold*, estão apresentados nas Tabelas 6, 7, 8 (Testes que não apresentam resultados indicam que, com aquela configuração, a base não possuía no mínimo 100 instâncias).

Tabela 6: Comparação entre os resultados obtidos nos testes do valor z = 64 na base de treinamento

Escala	Subfigura	Número Instâncias	Instâncias Corretas	Instâncias Erradas	Taxa de Acerto (%)	Taxa de Erro (%)
1000	100	692	680	12	98,2659	1,7341
1000	200	128	123	5	96,0938	3,9063
1000	300	-	-	-	-	-
1000	400	-	-	-	-	-
1500	100	1703	1660	43	97,4750	2,252
1500	200	367	363	4	98,9101	1,0899
1500	300	128	126	2	98,4375	1,5625
1500	400	-	-	-	-	-
2000	100	3159	3004	155	95,0934	4,9066
2000	200	692	676	16	97,6879	2,3121
2000	300	274	268	6	97,8102	2,1898
2000	400	128	125	3	97,6563	2,3438
2500	100	4971	4537	434	91,2694	8,7306
2500	200	1154	1117	37	96,7938	3,2062
2500	300	469	457	12	97,4414	2,5586
2500	400	246	242	4	98,374	1,626

Fonte: Autoria própria.

Tabela 7: Comparação entre os resultados obtidos nos testes do valor z = 96 na base de treinamento

Escala	Subfigura	Número Instâncias	Instâncias Corretas	Instâncias Erradas	Taxa de Acerto (%)	Taxa de Erro (%)
1000	100	692	677	15	97,8324	2,1676
1000	200	128	123	5	96,0938	3,9063
1000	300	-	-	-	-	-
1000	400	-	-	-	-	-
1500	100	1703	1652	51	97,0053	2,9947
1500	200	367	364	3	99,1826	0,8174
1500	300	128	125	3	97,6563	2,3438
1500	400	-	-	-	-	-
2000	100	3159	3001	158	94,9984	5,0016
2000	200	692	680	12	98,2659	1,7341
2000	300	274	272	2	99,2701	0,7299
2000	400	128	125	3	97,6563	2,3438
2500	100	4971	4554	417	91,6113	8,3887
2500	200	1154	1119	35	96,9671	3,2062
2500	300	469	458	11	97,6546	2,3454
2500	400	246	241	5	97,9675	2,0325

Fonte: Autoria própria.

Tabela 8: Comparação entre os resultados obtidos nos testes do valor z = 256 na base de treinamento

Escala	Subfigura	Número Instâncias	Instâncias Corretas	Instâncias Erradas	Taxa de Acerto (%)	Taxa de Erro (%)
1000	100	692	678	14	97,9769	2,0231
1000	200	128	123	5	96,0938	3,9063
1000	300	-	-	-	-	-
1000	400	-	-	-	-	-
1500	100	1703	1644	59	96,5355	3,4645
1500	200	367	364	3	99,1826	0,8174
1500	300	128	123	5	96,0938	3,9063
1500	400	-	-	-	-	-
2000	100	3159	2970	189	94,0171	5,9829
2000	200	692	683	9	98,6994	1,3006
2000	300	274	270	4	98,5401	1,4599
2000	400	128	124	4	96,875	3,125
2500	100	4971	4519	452	90,9073	9,0927
2500	200	1154	1121	33	97,1404	2,8596
2500	300	469	457	12	97,4414	2,5586
2500	400	246	241	5	97,9675	2,0325

Fonte: Autoria própria.

Além destes resultados, a Tabela 9 apresenta uma média de acerto e erro para cada valor de z. Percebe-se, a partir dela, que os resultados obtidos com valor de z igual à 64 e 96 são melhores, diferenciando-se minimamente. Contudo, a transformada de distância utilizada pelo método FD_{MIC} está em função da quantidade de camadas, que por sua vez está relacionada com a quantização, pois ela trabalha com uma matriz tridimensional e uma dessas dimensões é o z. Dessa forma, quanto menor este valor, menor o número de camadas e, por consequência, menor o custo computacional para execução do FD_{MIC}.

Tabela 9: Média de acerto e erro dos testes de quantização no estudo de caso

Valor z	Média de Acerto (%)	Média de Erro (%)
64	97,0237	2,9763
96	97,0893	2,9107
256	96,7285	3,2715

Fonte: Autoria própria.

Sendo assim, decidiu-se utilizar para o treinamento do modelo: valor de z igual à 64, valor da escala igual à 1000 e valor da subimagem igual à 100, pois este conjunto de configurações alinha uma boa acurácia com eficiência de execução, tornando-se apropriado para o uso em dispositivos móveis que, naturalmente, possuem um poder de processamento reduzido.

Definido os valores de escala e subimagem, consequentemente define-se qual o tamanho das bases, de treinamento e teste, que serão geradas, bem como a quantidade de imagens de cada espécie contidas nelas, conforme mostra a Tabela 10 e 11, respectivamente. A Figura 14 apresenta amostras da base de treinamento, enquanto a Figura 15 apresenta amostras da base de teste.

Tabela 10: Relação entre classe de textura e seu respectivo número de instâncias para a base de treinamento do estudo de caso

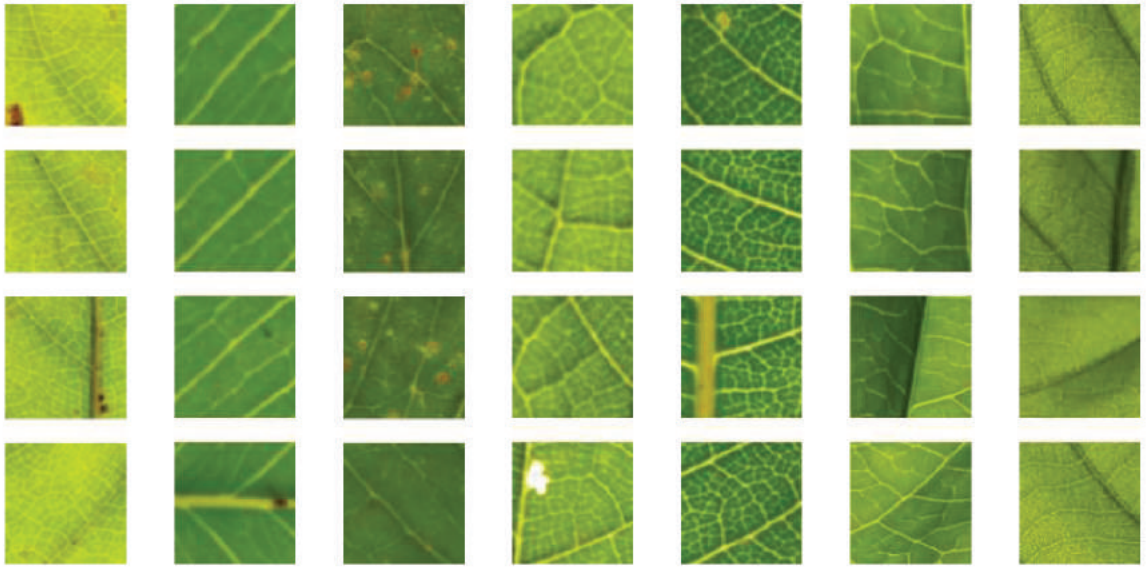
Classes	Quantidade
Abacate	159
Araça Amarelo	27
Araça Vermelho	54
Chuchu	149
Manga	84
Maracujá	122
Nêspera	97
TOTAL	**692**

Fonte: Autoria própria.

Tabela 11: Relação entre classe de textura e seu respectivo número de instâncias para a base de teste do estudo de caso

Classes	Quantidade
Abacate	318
Araça Amarelo	67
Araça Vermelho	169
Chuchu	246
Manga	245
Maracujá	120
Nêspera	206
TOTAL	**1.371**

Fonte: Autoria própria.



(a) (b) (c) (d) (e) (f) (g)

Figura 14: Amostras de imagens contidas na base de treinamento das plantas. Na qual: (a) Abacate, (b) Araça Amarelo, (c) Araça Vermelho, (d) Chuchu, (e) Manga, (f) Maracujá e (g) Nêspera.

Fonte: Autoria Própria

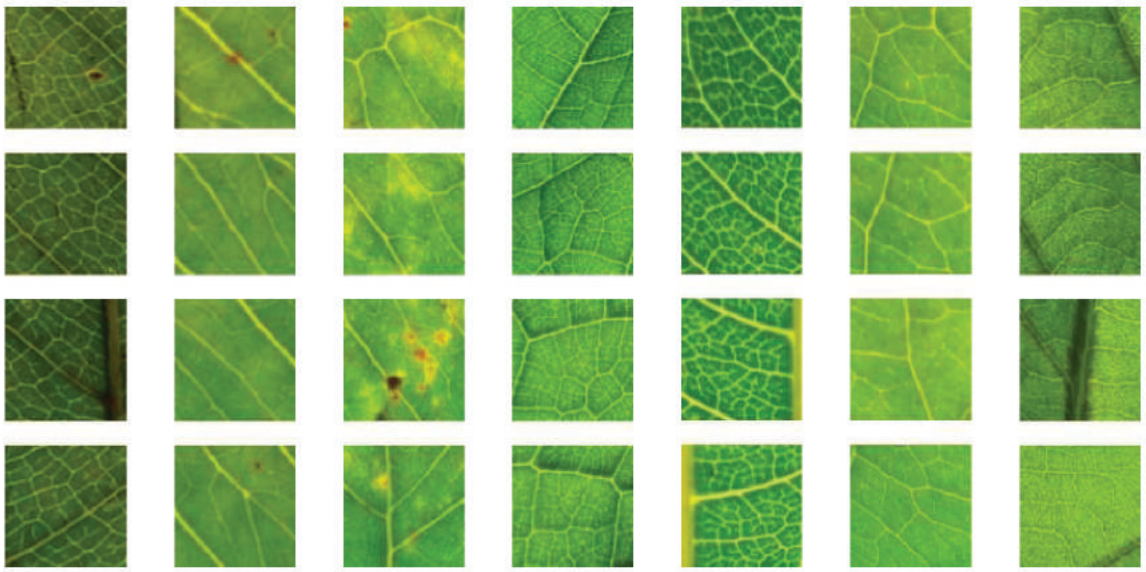


Figura 15: Amostras de imagens contidas na base de teste das plantas. Na qual: (a) Abacate, (b) Araça Amarelo, (c) Araça Vermelho, (d) Chuchu, (e) Manga, (f) Maracujá e (g) Nêspera.

Fonte: Autoria Própria

6.2.3 RESULTADOS DO ESTUDO DE CASO

Juntamente com a geração das bases, treinou-se um modelo classificador a partir da base de treinamento e criou-se um arquivo ARFF a partir da base de teste. Com estes dois artefatos foi possível medir a acurácia do modelo por meio do próprio software do Weka, simulando a utilização deste modelo no aplicativo móvel. Diferentemente dos testes anteriores, que utilizaram validação cruzada, o resultado deste foi obtido carregando o modelo ao Weka e fornecendo o arquivo ARFF como o conjunto de teste, os dados estatísticos obtidos estão apresentados na Tabela 12.

Tabela 12: Resultado do teste de acurácia sobre o modelo classificador do estudo de caso

Número Instâncias	Instâncias Corretas	Instâncias Erradas	Taxa de Acerto (%)	Taxa de Erro (%)
1371	446	925	32,531	67,469

Fonte: Autoria própria.

A partir do resultado, constata-se que utilizando duas bases coloridas capturadas por dois dispositivos diferentes a taxa de acerto é baixa, indicando que a base de treinamento não é representativa para este caso, necessitando um número maior de amostras e amostras capturadas por diferentes dispositivos, para que o modelo seja treinado com uma diversidade maior de variações de variáveis.

Após a análise sobre as bases utilizadas, notou-se uma nítida diferença apresentada em relação à coloração das imagens de uma mesma espécie, que pode ser percebida comparando as Figuras 14 e 15, o que enfatiza a não representatividade da base de treinamento. Além disso, esta diferenciação interfere diretamente no método FD_{MIC}, que é suscetível a variações de cor, sendo assim imagens de uma textura, mas com distorção na coloração, são classificadas como classes distintas, sendo este, um dos motivos pela baixa taxa de acerto.

Deste modo, realizou-se mais três testes: isolamento de variáveis (Seção 6.3), conversão das imagens em tons de cinza (Seção 6.4) e isolamento do *smartphone* (Seção 6.5).

6.3 TESTE DE ISOLAMENTO DE VARIÁVEIS

Esta seção descreve o experimento de isolar variáveis no momento da aquisição das imagens de treinamento e teste, tais como: angulação, distância e iluminação, mantendo variável somente o processamento da câmera dos próprios *smartphones*. Isto pois, estas variáveis podem influenciar diretamente no método FD_{MIC} ao extrair as características.

Seguindo a mesma metodologia apresentada na Seção 6.2 do estudo de caso, utilizou-se dois dispositivos diferentes que são apresentados no Quadro 3, juntamente com as dimensões das imagens capturadas por ambos.

Dispositivo	**Dimensão (A x L)**	**Base**
Motorola Moto G5S	4632 x 3474	Treinamento
Xiaomi Mi A2 Lite	4000 x 3000	Teste

Quadro 3: Informações dos modelos de *smartphones* utilizados na captura de imagens no teste de isolamento de variáveis

Fonte: Autoria própria.

Para este teste manteve-se o valor de z fixo em 64, bem como a escala e o tamanho da subimagem em 1000 e 100, respectivamente. Além disso, criou-se uma nova base de treino e outra de teste com diferentes classes, capturando texturas diversas no câmpus da UTFPR em Dois Vizinhos-PR, conforme descritas nas Tabelas 13 e 14, respectivamente, descrevendo as bases de treino e teste. A Figura 16 apresenta amostras da base de treinamento, enquanto a Figura 17 apresenta amostras da base de teste.

Tabela 13: Relação entre classe de textura e seu respectivo número de instâncias para a base de treinamento no isolamento de variáveis

Classes	Quantidade
Árvore	336
Grama	306
Paralelepípedo	294
Parede	350
Paver (1)	334
Paver (2)	350
Piso de concreto	320
Porta de metal	350
Parede de Granito	350
TOTAL	**2.990**

Fonte: Autoria própria.

Tabela 14: Relação entre classe de textura e seu respectivo número de instâncias para a base de teste no isolamento de variáveis

Classes	Quantidade
Árvore	350
Grama	250
Paralelepípedo	340
Parede	350
Paver (1)	343
Paver (2)	350
Piso de concreto	280
Porta de metal	350
Parede de Granito	350
TOTAL	**2.963**

Fonte: Autoria própria.

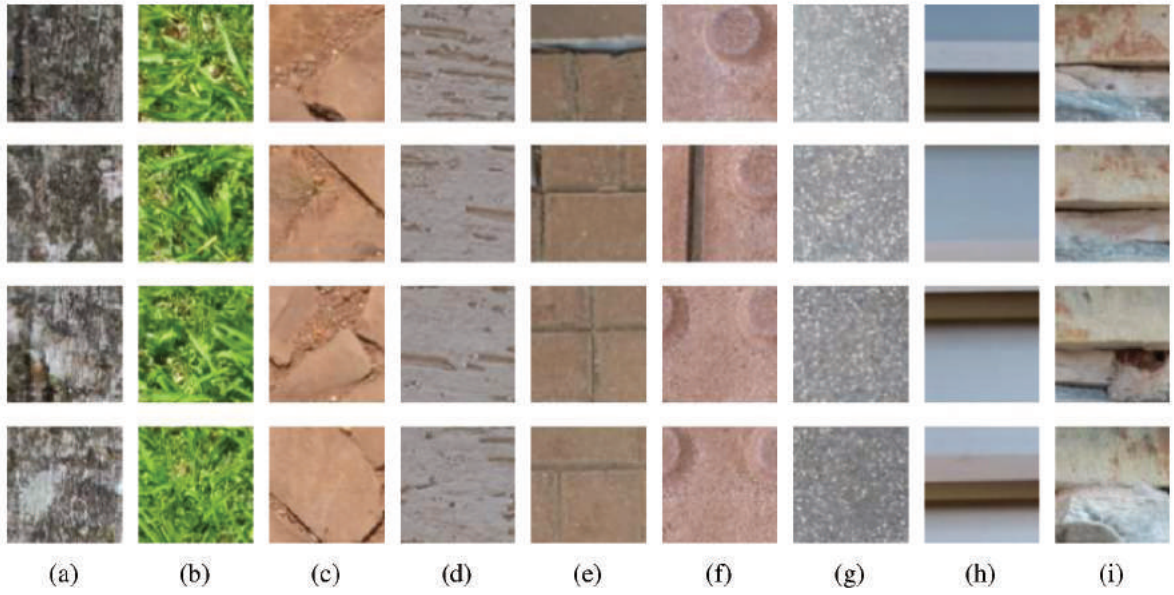


Figura 16: Amostras de imagens contidas na base de treino das texturas diversas. Na qual: (a) Árvore, (b) Grama, (c) Paralelepípedo, (d) Parede, (e) Paver (1), (f) Paver (2), (g) Piso de Concreto, (h) Porta de Metal e (i) Parede de Granito.

Fonte: Autoria Própria

Figura 17: Amostras de imagens contidas na base de teste das texturas diversas. Na qual: (a) Árvore, (b) Grama, (c) Paralelepípedo, (d) Parede, (e) Paver (1), (f) Paver (2), (g) Piso de Concreto, (h) Porta de Metal e (i) Parede de Granito.

Fonte: Autoria Própria

O resultado, apresentado na Tabela 15, foi obtido carregando o modelo treinado no Weka e fornecendo o arquivo ARFF como o conjunto de teste. Em comparação com o resultado do estudo de caso (apresentado na Seção 6.2.3), este apresentou significativa melhora na taxa de acerto, tendo um aumento de aproximadamente 44,553%.

Tabela 15: Resultado do teste de acurácia sobre o modelo classificador isolando variáveis

Número Instâncias	Instâncias Corretas	Instâncias Erradas	Taxa de Acerto (%)	Taxa de Erro (%)
2963	2284	679	77,084	22,916

Fonte: Autoria própria.

Ainda realizou-se um teste, cujo resultado é apresentado na Tabela 16, no qual uniu-se as duas bases de texturas diversas capturadas pelos dois dispositivos e obteve-se os valores estatísticos por meio de validação cruzada *10-fold*. A partir dele, nota-se que mesmo o teste anterior tendo registrado uma aumento na taxa de acerto em relação ao estudo de caso, a acurácia do classificador utilizando dois *smartphones* distintos é distante do valor obtido por validação cruzada, no qual a base de treinamento é altamente representativa.

Tabela 16: Resultado do teste de acurácia isolando variáveis obtido por meio da validação cruzada

Número Instâncias	Instâncias Corretas	Instâncias Erradas	Taxa de Acerto (%)	Taxa de Erro (%)
5953	5811	142	97,6146	2,3854

Fonte: Autoria própria.

Além disso, esta melhora pode estar relacionada não somente com o isolamento das variáveis, mas também com uma nítida diferença de características entre as próprias classes pertencentes à base de texturas diversas. Ao contrário da base de plantas, na qual as classes pertencentes à ela possuem características mais semelhantes, dificultando a classificação das espécies. Isto porque, analisando as bases utilizadas neste teste, mesmo isolando as variáveis e igualmente ao ocorrido nas bases do estudo de caso, estas apresentam o mesmo problema de diferenças na coloração entre uma mesma classe, que pode ser percebida comparando as Figuras 16 e 17.

Portanto, pode-se sugerir que a taxa de acerto em relação à base de plantas aumentou devido ao isolamento das variáveis e a nítida diferença de características entre as classes da base de textura, mas não foi melhor por conta da diferenciação de cor entre as mesmas classes.

6.4 TESTE DAS IMAGENS EM TONS DE CINZA

Esta seção detalha o teste realizado com o objetivo de avaliar a acurácia do método FD_{MIC} sobre as bases de plantas e texturas diversas, convertendo as imagens para tons de cinza, visto que os resultados obtidos ao utilizar estas mesmas bases coloridas (Seções 6.2.3 e 6.3) não foram os desejáveis.

Para tanto, foram utilizadas as bases de treinamento e de teste já descritas anteriormente, respectivamente, nas Tabelas 10 e 11, referente à base de plantas e nas Tabelas 13 e 14, referente à base de texturas diversas. Contudo, tornou-se necessário converter as imagens contidas nelas para tons de cinza imediatamente antes de realizar a extração de características. O resultado da conversão pode ser visualizado nas Figuras 18 e 19, que apresentam amostras em tons de cinza da base de plantas e nas Figuras 20 e 21 que apresentam amostras da base de texturas diversas.

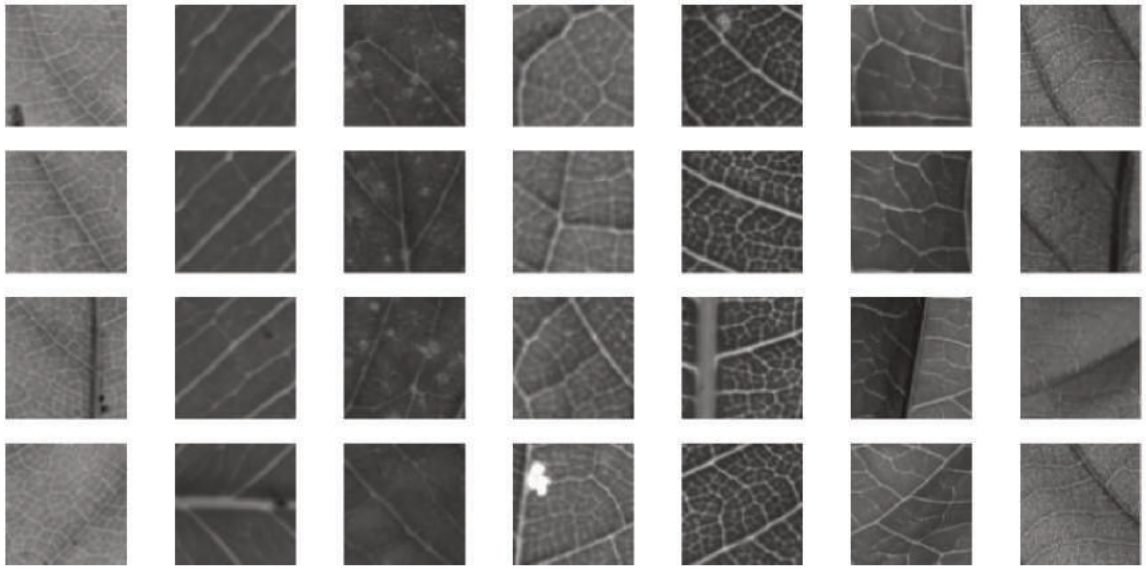


Figura 18: Amostras de imagens contidas na base de treinamento das plantas em tons de cinza. Na qual: (a) Abacate, (b) Araça Amarelo, (c) Araça Vermelho, (d) Chuchu, (e) Manga, (f) Maracujá e (g) Nêspera.

Fonte: Autoria Própria

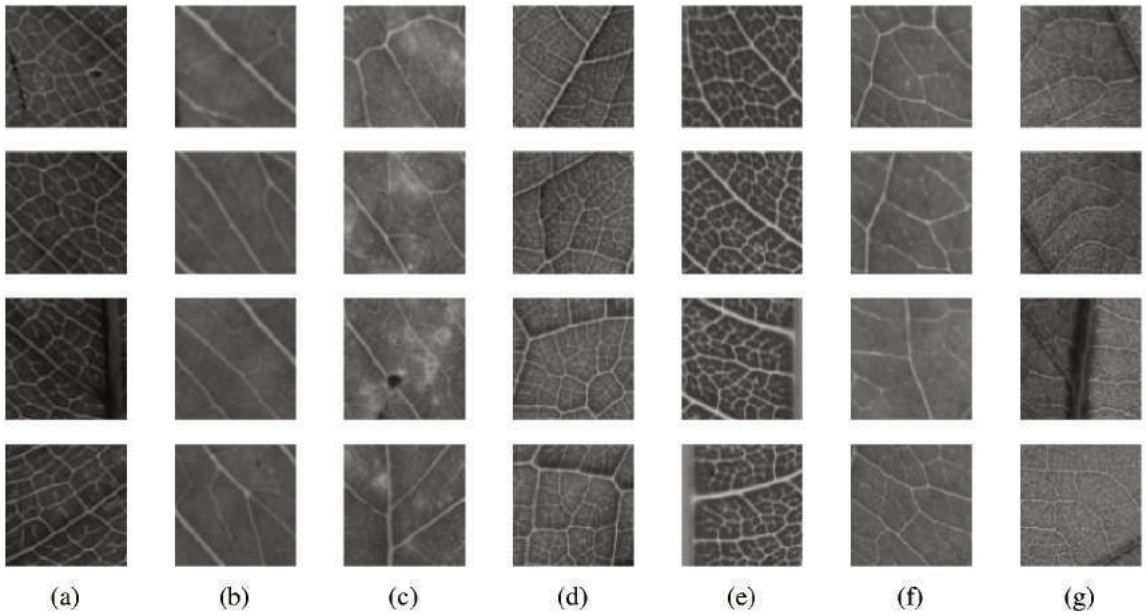


Figura 19: Amostras de imagens contidas na base de teste das plantas em tons de cinza. Na qual: (a) Abacate, (b) Araça Amarelo, (c) Araça Vermelho, (d) Chuchu, (e) Manga, (f) Maracujá e (g) Nêspera.

Fonte: Autoria Própria

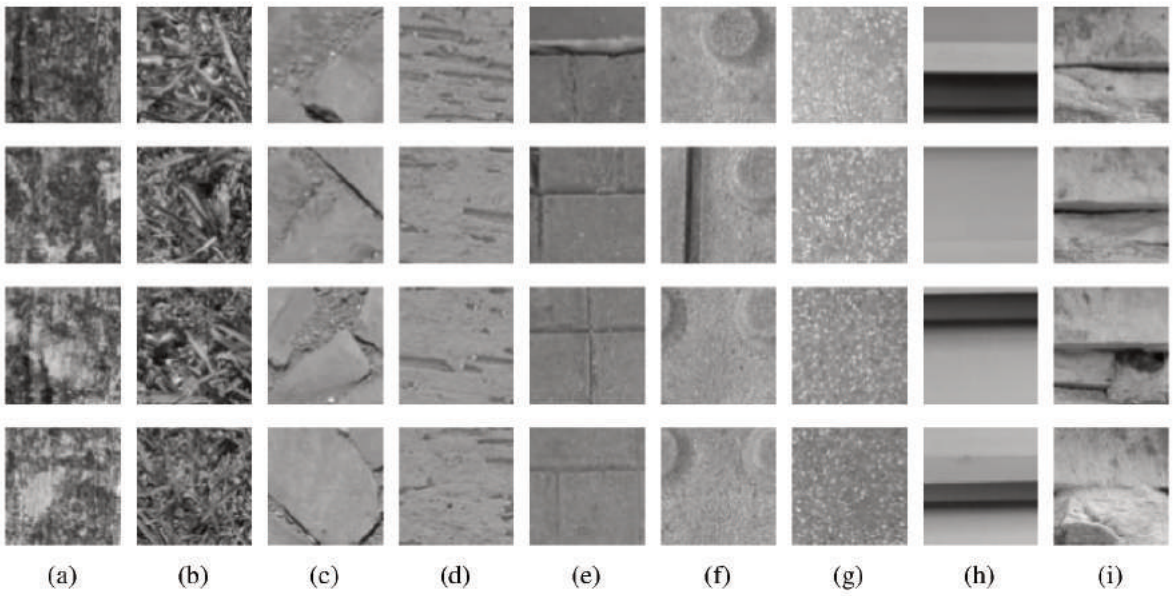


Figura 20: Amostras de imagens contidas na base de treino das texturas diversas em tons de cinza. Na qual: (a) Árvore, (b) Grama, (c) Paralelepípedo, (d) Parede, (e) Paver (1), (f) Paver (2), (g) Piso de Concreto, (h) Porta de Metal e (i) Parede de Granito.

Fonte: Autoria Própria

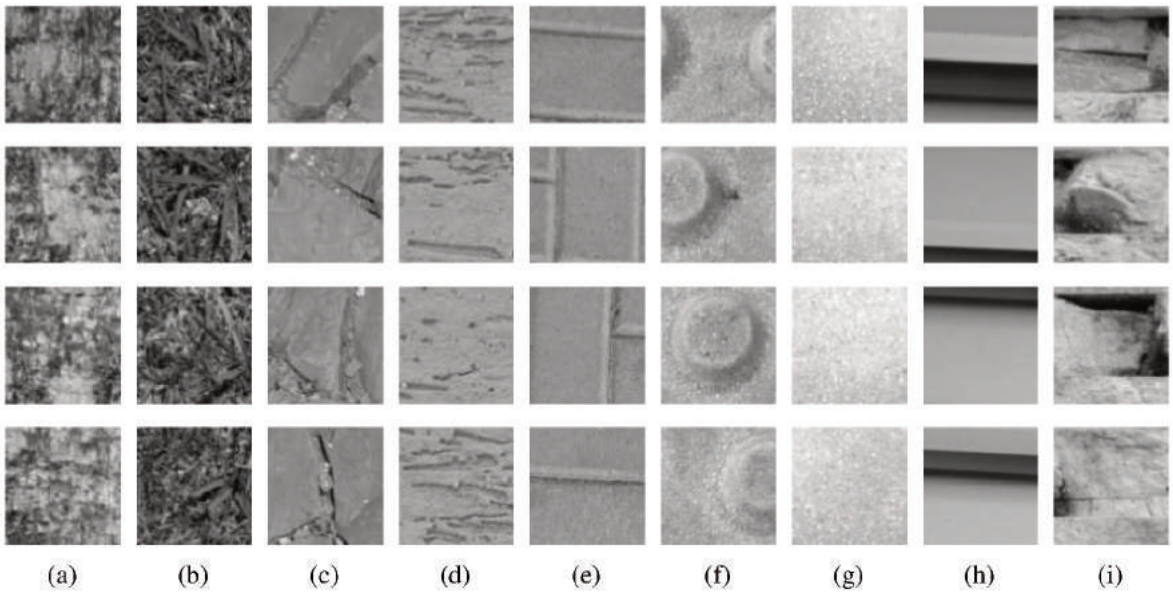


Figura 21: Amostras de imagens contidas na base de teste das texturas diversas em tons de cinza. Na qual: (a) Árvore, (b) Grama, (c) Paralelepípedo, (d) Parede, (e) Paver (1), (f) Paver (2), (g) Piso de Concreto, (h) Porta de Metal e (i) Parede de Granito.

Fonte: Autoria Própria

Utilizando as mesmas bases, indiretamente, seguiu-se as mesmas configurações adotadas nos testes anteriores, ou seja, manteve-se o valor de z fixo em 64, a escala em 1000

e o tamanho da subimagem em 100. O resultado, que pode ser visualizados na Tabela 17, foi obtidos carregando o modelo no Weka e fornecendo o arquivo ARRF como conjunto de teste.

Tabela 17: Resultado dos testes com as bases em tons de cinza

Base	Número Instâncias	Instâncias Corretas	Instâncias Erradas	Taxa de Acerto (%)	Taxa de Erro (%)
Plantas	1371	775	596	56,5281	43,4719
Texturas Diversas	2963	2651	312	89,4701	10,5299

Fonte: Autoria própria.

A partir da análise do resultado, constata-se que a taxa de acerto, em relação a base de plantas, com imagens em tons de cinza apresentou um aumento de, aproximadamente, 24,00% em comparação com a base colorida. Se tratando da base de texturas diversas em tons de cinza, o aumento da taxa de acerto foi de, aproximadamente, 12,00% em relação à base colorida.

Como constatado em testes anteriores, imagens de uma mesma textura capturadas por dispositivos diferentes, apresentam distorções nas cores e como o método FD_{MIC} é variável a coloração ele interpreta que estas imagens pertencem a classes distintas. Neste contexto, o resultado obtido neste teste sugere que eliminando a variação de cores o método de extração de características FD_{MIC} torna-se mais acurado.

Contudo, devido a distorção de cores nas imagens originais da base de plantas, ao convertê-las para tons de cinza, a distorção, agora em níveis de cinza, permaneceu, como pode ser comparado entre Figuras 18 e 19. Portanto, mesmo com o considerável aumento descrito anteriormente, a taxa de acerto obtida para esta base não é satisfatória.

6.5 TESTE ISOLAMENTO DO *SMARTPHONE*

Esta seção detalha o experimento isolando o *smartphone*, ou seja, as imagens capturadas pelo mesmo dispositivo foram divididas em base de treinamento e base de teste, visando medir e comparar a acurácia com os resultados obtidos nos testes das Seções 6.2 e 6.3, os quais utilizaram dois dispositivos distintos.

Esta divisão das bases capturadas por um mesmo dispositivos difere-se da divisão realizada pela validação cruzada em testes anteriores. Na validação cruzada, há possibilidade de que subimagens de uma mesma imagem estejam contidas tanto na base de treinamento quanto na base de teste. Neste caso, eliminou-se esta possibilidade de ligação trabalhando com cada base individualmente e controlando o processo de divisão das imagens. Sendo assim, tanto o modelo a ser carregado no Weka quanto o arquivo ARFF para conjunto de teste, serão gerados

por imagens capturadas pelo mesmo dispositivo, processo diferente de testes anteriores que utilizavam um dispositivo para gerar cada artefato.

Para o experimento, utilizou-se as duas bases coloridas já coletadas (plantas e texturas diversas), dessa maneira foi necessário dividir as imagens capturadas por cada dispositivo, sendo eles: Motorola Moto G5S, Xiaomi Mi A2 Lite e Asus Zenfone 5.

As Tabelas 18 e 19 apresentam informações, respectivamente, à respeito das bases de treinamento e teste das plantas capturadas pelo dispositivo Moto G5S, enquanto as Tabelas 20 e 21 apresentam as mesmas informações do dispositivo Asus Zenfone 5. As Tabelas 22 e 23 apresentam informações sobre as bases de texturas diversas capturadas pelo Moto G5S, enquanto as Tabelas 24 e 25 apresentam as mesmas informações do Mi A2 Lite.

Tabela 18: Relação entre classe de textura e seu respectivo número de instâncias para a base de treinamento das plantas isolando o dispositivo Moto G5S

Classes	Quantidade
Abacate	100
Araça Amarelo	18
Araça Vermelho	30
Chuchu	100
Manga	52
Maracujá	62
Nêspera	58
TOTAL	**420**

Fonte: Autoria própria.

Tabela 19: Relação entre classe de textura e seu respectivo número de instâncias para a base de teste das plantas isolando o dispositivo Moto G5S

Classes	Quantidade
Abacate	59
Araça Amarelo	9
Araça Vermelho	24
Chuchu	49
Manga	32
Maracujá	60
Nêspera	39
TOTAL	**272**

Fonte: Autoria própria.

Tabela 20: Relação entre classe de textura e seu respectivo número de instâncias para a base de treinamento das plantas isolando o dispositivo Zenfone 5

Classes	Quantidade
Abacate	178
Araça Amarelo	40
Araça Vermelho	96
Chuchu	160
Manga	160
Maracujá	81
Nêspera	126
TOTAL	**841**

Fonte: Autoria própria.

Tabela 21: Relação entre classe de textura e seu respectivo número de instâncias para a base de teste das plantas isolando o dispositivo Zenfone 5

Classes	Quantidade
Abacate	140
Araça Amarelo	27
Araça Vermelho	73
Chuchu	86
Manga	85
Maracujá	39
Nêspera	80
TOTAL	**530**

Fonte: Autoria própria.

Tabela 22: Relação entre classe de textura e seu respectivo número de instâncias para a base de treinamento das texturas diversas isolando o dispositivo Moto G5S

Classes	Quantidade
Árvore	203
Grama	180
Paralelepípedo	184
Parede	210
Paver (1)	194
Paver (2)	210
Piso de concreto	180
Porta de metal	210
Parede de Granito	210
TOTAL	**1.781**

Fonte: Autoria própria.

Tabela 23: Relação entre classe de textura e seu respectivo número de instâncias para a base de teste das texturas diversas isolando o dispositivo Moto G5S

Classes	Quantidade
Árvore	133
Grama	126
Paralelepípedo	110
Parede	140
Paver (1)	140
Paver (2)	140
Piso de concreto	140
Porta de metal	140
Parede de Granito	140
TOTAL	**1.209**

Fonte: Autoria própria.

Tabela 24: Relação entre classe de textura e seu respectivo número de instâncias para a base de treinamento das texturas diversas isolando o dispositivo Mi A2 Lite

Classes	Quantidade
Árvore	210
Grama	130
Paralelepípedo	210
Parede	210
Paver (1)	210
Paver (2)	210
Piso de concreto	161
Porta de metal	210
Parede de Granito	210
TOTAL	**1.761**

Fonte: Autoria própria.

Tabela 25: Relação entre classe de textura e seu respectivo número de instâncias para a base de teste das texturas diversas isolando o dispositivo Mi A2 Lite

Classes	Quantidade
Árvore	140
Grama	120
Paralelepípedo	130
Parede	140
Paver (1)	133
Paver (2)	140
Piso de concreto	119
Porta de metal	140
Parede de Granito	140
TOTAL	**1.202**

Fonte: Autoria própria.

Para este, manteve-se as configurações de testes anteriores: o valor de z permaneceu fixo em 64, a escala em 1000 e o tamanho da subimagem em 100. O resultado é apresentado na Tabela 26.

Tabela 26: Resultado dos testes isolando os *smartphones*

Base	Dispositivo	Número Instâncias	Instâncias Corretas	Instâncias Erradas	Taxa de Acerto (%)	Taxa de Erro (%)
Plantas	Moto G5S	272	246	26	90,4412	9,5588
Plantas	Zenfone 5	530	463	67	87,3585	12,6415
Texturas Diversas	Moto G5S	1209	1100	109	90,9843	9,0157
Texturas Diversas	Mi A2 Lite	1202	1179	23	98,0865	1,9135

Fonte: Autoria própria.

Este teste apresentou resultados melhores em relação à testes anteriores, sendo as taxas de acerto acima de 87,00%. A partir da análise, é possível constatar que a utilização de imagens capturadas pelo mesmo dispositivo, para treinar e classificar, apresenta uma taxa de acerto maior em comparação com a utilização de um dispositivo distinto para cada uma das etapas.

Esta diferenciação decorre, principalmente, do próprio processamento e tratamento realizado pelo dispositivo no momento da captura. Ao utilizar dois dispositivos distintos ocorrem distorções nas imagens de treinamento e teste obtidas por cada um deles, entre elas a própria cor, o que resulta em uma baixa taxa de acerto. Ao utilizar apenas um dispositivo, e dessa maneira isolando as variáveis particulares de cada um, evita-se que estas distorções ocorram de forma acentuada nas imagens, assim aumenta-se a taxa de acerto.

No entanto, cada *smartphone* possui funcionamento e configurações particulares que podem não estar sob o controle do usuário, desse modo apresenta um risco ao treinamento mesmo utilizando um único aparelho. Enquanto que configurações sob controle do usuário tais como: modo HDR, flash e resolução, precisam se manter fixas para a captura das imagens de treinamento, caso contrário, do mesmo modo, apresentarão riscos.

7 CONSIDERAÇÕES FINAIS

Este trabalho propôs o desenvolvimento de uma aplicação móvel capaz de classificar imagens de textura, utilizando o recente método extrator de características FD_{MIC}. A abordagem consiste em gerar um modelo classificador por meio de um software *desktop* e então integrá-lo ao aplicativo para classificar alguma imagem de textura. Sendo esta, uma tentativa de contribuir à literatura com uma aplicação prática utilizando FD_{MIC}.

Os resultados obtidos no estudo de caso (Seção 6.2) não se mostraram satisfatórios, já que a proposta visava criar um modelo treinado a partir de imagens capturadas por um único dispositivo e que pudesse ser utilizado na aplicação instalada em qualquer dispositivo diferente. O principal fator decisivo está relacionado com o próprio método de processamento de imagens de cada dispositivo, os quais diferem entre si. Testes descritos nas Seções 6.3 e 6.4 reforçam esta hipótese, pois também utilizaram dois dispositivos distintos no treinamento e teste do modelo.

Este risco não foi pontuado na idealização do projeto, mostrando-se determinante nos resultados. Contudo, o teste descrito na Seção 6.5 obteve resultados satisfatórios, indicando que o uso da aplicação utilizando imagens referentes ao mesmo dispositivo é altamente viável, pois elimina a variável de processamento de cada dispositivos, apresentando boas taxas de acerto.

Além do desenvolvimento de uma aplicação móvel em Android, integrando bibliotecas de processamento de imagem e inteligência artificial, e o desenvolvimento de um software *desktop*, escrito em Java, capaz de criar um modelo classificador a partir de uma base de imagens fornecida, este trabalho também contribuiu com:

- Implementação do método FD_{MIC} na linguagem Java (Disponível em: https://github.com/feliperpv/analise-textura-fdmic-java);
- Criação de uma base de imagens de texturas, com 2.063 imagens 100 x 100, contendo 7 classes diferentes de plantas (Disponível em: https://goo.gl/ZgkFAp);
- Criação de uma base de imagens de texturas diversas, com 5.953 imagens 100 x 100 contendo 9 classes (Disponível em: https://goo.gl/EfQQHp).

Além dos resultados e contribuições apresentadas, identifica-se trabalhos futuros relacionados a esta proposta, que podem contribuir com os resultados obtidos e expandir a aplicação desenvolvida:

- Desenvolvimento de um software que dado a imagem de um fundo padrão, detecta as distorções ocasionadas pelo processamento dos dispositivos e as desfaça-as. Desse modo as imagens seriam normalizadas, reduzindo o impacto das distorções entre imagens de uma mesma classe nos resultados;
- Expandir o alcance da aplicação, desenvolvendo, com os devidos ajustes, para dispositivos que utilizem outros sistemas operacionais;
- Desenvolvimento de um mecanismo ou ferramenta que faça a integração dos arquivos gerados pelo software *desktop* com a aplicação móvel de forma automática.

REFERÊNCIAS

BACKES, A. R. **Implementação e comparação de métodos de estimativa da dimensão fractal e sua aplicação à análise e processamento de imagens**. 2006. Dissertação (Mestrado em Ciências de Computação e Matemática Computacional) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2006.

BMSVISION. **Cyclops - Automatic on loom fabric inspection**. 2018. Acesso em: 17 maio 2018. Disponível em: <http://www.bmsvision.com/en/products/cyclops>.

CAI, D.; HE, X.; HAN, J. Training linear discriminant analysis in linear time. **2008 IEEE 24th International Conference on Data Engineering (ICDE)**, p. 209–217, 2008.

CASANOVA, D. et al. Texture analysis using fractal descriptors estimated by the mutual interference of color channels. **Information Sciences**, v. 346-347, p. 58–72, jun. 2016.

CHAUDHARI, S. I. et al. A survey on detection of unhealthy region of plant leaves by using image processing. **2016 International Research Journal of Engineering and Technology (IRJET)**, v. 03, n. 09, sep. 2016.

DEVORE, J. L. **Probability and Statistics for Engineering and the Sciences**. 8th. ed. [S.l.]: Brooks/Cole, 2011.

FISHER, R. A. The use of multiple measurements in taxonomic problems. **Annals of Eugenics**, v. 7, p. 179–188, 1936.

FLORINDO, J. B. **Descritores fractais aplicados à análise de texturas**. 2013. Tese (Doutorado em Física Aplicada) - Instituto de Física de São Carlos, Universidade de São Paulo, São Carlos, 2016.

GOëAU, H. et al. Pl@ntnet mobile 2014: Android port and new features. **2014 International Conference on Multimedia Retrieval (ICMR)**, p. 527–528, abr. 2014.

GOLAN, J. S. **The linear algebra a beginning graduate student ought to know**. 3. ed. [S.l.]: Springer, 2012.

GOYAL, A. 4 - automation in fabric inspection. In: NAYAK, R.; PADHYE, R. (Ed.). **Automation in Garment Manufacturing**. Woodhead Publishing, 2018, (The Textile Institute Book Series). p. 75–107. Disponível em: <http://www.sciencedirect.com/science/article/pii/B9780081012116000045>.

GUNASEKARA, P. G. H. H.; WIJAYAKULASOORIYA, J. V.; DHARMAGUNAWARDHANA, H. A. C. Image texture analysis using deep neural networks. **2017 IEEE International Conference on Industrial and Information Systems (ICIIS)**, p. 1–5, dez. 2017.

GUO, Y.; HASTIE, T.; TIBSHIRANI, R. Regularized linear discriminant analysis and its application in microarrays. **Biostatistics**, v. 8, n. 1, p. 86–100, jan. 2007.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The elements of statistical learning: data mining, inference and prediction**. 2. ed. Berlim: Springer, 2009.

ISLAM, M. N. et al. Skin disease recognition using texture analysis. **2017 IEEE 8th Control and System Graduate Research Colloquium (ICSGRC)**, p. 144–148, aug. 2017.

JOLY, A. et al. Interactive plant identification based on social image data. **Ecological Informatics**, v. 23, p. 22–34, set. 2014.

KHADEMI, A.; KRISHNAN, S. Medical image texture analysis: A case study with small bowel, retinal and mammogram images. **2008 Canadian Conference on Electrical and Computer Engineering**, p. 001949–001954, maio 2008.

KUMAR, A. Computer-vision-based fabric defect detection: A survey. **IEEE Transactions on Industrial Electronics**, v. 55, n. 1, p. 348–363, jan. 2008.

MA, L. et al. Personal identification based on iris texture analysis. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 25, n. 12, p. 1519–1533, dez. 2003.

MAHAJAN, P.; KOLHE, S.; PATIL, P. A review of automatic fabric defect detection techniques. **Advances in Computational Research**, v. 1, jan. 2009.

MALAMAS, E. N. et al. A survey on industrial vision systems, applications and tools. **Image and Vision Computing**, v. 21, n. 2, p. 171–188, fev. 2003.

MARASCA, A.; CASANOVA, D.; TEIXEIRA, M. Assessing classification complexity of datasets using fractals. **Submetido ao International Journal of Computational Science and Engineering**, 2018. Aceito para publicação.

MARASCA, A. L. **Análise de textura através de descritores fractais por meio da utilização de rótulos na transformada de distância 3D**. 2016. 72 f. Trabalho de Conclusão de Curso (Graduação) - Universidade Tecnológica Federal do Paraná, Pato Branco, 2016.

MATHWORKS. **Texture Analysis**. 2018. Disponível em: <https://www.mathworks.com/help/images/texture-analysis.html>. Acesso em: 14 mar. 2018.

MIT, M. L. **VisTex vision texture Database**. 1995. Disponível em: <http://vismod.media.mit.edu/vismod/imagery/VisionTexture/vistex.html>. Acesso em: 25 maio 2018.

NGAN, H. Y.; PANG, G. K.; YUNG, N. H. Automated fabric defect detection — a review. **Image and Vision Computing**, v. 29, n. 7, p. 442–458, jun. 2011.

OJALA, T. et al. Outex - new framework for empirical evaluation of texture analysis algorithms. **Object recognition supported by user interaction for service robots**, v. 1, p. 701–706, jan. 2002.

PRASAD, S.; KUMAR, P.; TRIPATHI, R. C. Plant leaf species identification using curvelet transform. **2011 2nd International Conference on Computer and Communication Technology (ICCCT)**, p. 646–652, set. 2011.

RASCHKA, S. **Python Machine Learning**. [S.l.]: Packt Publishing, 2015.

RODRIGUEZ, J. D.; PEREZ, A.; LOZANO, J. A. Sensitivity analysis of k-fold cross validation in prediction error estimation. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 32, n. 3, p. 569–575, mar. 2010.

SAITO, T.; TORIWAKI, J. New algorithms for euclidean distance transformation of an n-dimensional digitised picture with applications. v. 27, n. 11, 1994.

SHALABI, L.; ZYAD, S.; AL-KASASBEH, B. Data mining: A preprocessing engine. **Journal of Computes Science**, v. 2, set. 2006.

SHALEV-SHWARTZ, S.; BEN-DAVID, S. **Understanding Machine Learning: From theory to algorithms**. Nova Iorque: Cambridge University Press, 2014.

SOLEM, J. E. **Programming Computer Vision with Python**. [S.l.]: O'Reilly Media, 2012. 247 p.

SZCZYPIŃSKI, P. M.; KLEPACZKO, A. Chapter 11 - mazda – a framework for biomedical image texture analysis and data exploration. In: DEPEURSINGE, A.; AL-KADI, O. S.; MITCHELL, J. (Ed.). **Biomedical Texture Analysis**. [S.l.]: Academic Press, 2017. p. 315–347.

SZCZYPIŃSKI, P. M. et al. Mazda—a software package for image texture analysis. **Computer Methods and Programs in Biomedicine**, v. 94, n. 1, p. 66–76, 2009.

VISA, S. et al. Confusion matrix-based feature selection. **2011 Midwest Artificial Intelligence and Cognitive Science Conference**, v. 710, p. 120–127, 2011.

WÄLDCHEN, J.; MÄDER, P. Plant species identification using computer vision techniques: A systematic literature review. **Archives of Computational Methods in Engineering**, v. 25, n. 2, p. 507–543, abr. 2018.

WÄLDCHEN, J. et al. Automated plant species identification - trends and future directions. **PLOS Computational Biology**, Public Library of Science, v. 14, n. 4, p. 1–19, abr. 2018.

WALPOLE, R. E. et al. **Probability & Statistics for engineers and scientists**. 8th. ed. Upper Saddle River: Pearson Education, 2007.

WITTEN, I. H.; FRANK, E.; HALL, M. A. **Data Mining: Practical Machine Learning Tools and Techniques**. 4. ed. Amsterdam: Morgan Kaufmann, 2016.