UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO DE CIÊNCIAS, TECNOLOGIAS E SAÚDE
(CTS)

Luiz Antonio Buschetto Macarini

SISTEMA DE DETECÇÃO DE DEFEITOS VISUAIS EM PISOS CERÂMICOS BASEADO EM PROCESSAMENTO DE IMAGENS E APRENDIZADO DE MÁQUINA

Araranguá

2017

Luiz Antonio Buschetto Macarini

SISTEMA DE DETECÇÃO DE DEFEITOS VISUAIS EM PISOS CERÂMICOS BASEADO EM PROCESSAMENTO DE IMAGENS E APRENDIZADO DE MÁQUINA

Trabalho de Conclusão de Curso submetido à Universidade Federal de Santa Catarina para a obtenção do Grau de Bacharel em Engenharia de Computação. Orientador: Prof. Dr. Tiago Oliveira Weber.

Araranguá

2017

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Macarini, Luiz Antonio Buschetto
Sistema de Detecção de Defeitos Visuais em Pisos Cerâmicos baseado em Processamento de Imagens e Aprendizado de Máquina / Luiz Antonio Buschetto Macarini ; orientador, Tiago Oliveira Weber, 2017.
106 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Campus Araranguá, Graduação em Engenharia de Computação, Araranguá, 2017.

Inclui referências.

1. Engenharia de Computação. 2. Processamento de Imagens. 3. Aprendizado de Máquina. 4. Visão de Máquina. 5. Controle de Qualidade. I. Weber, Tiago Oliveira. II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Computação. III. Título.

Luiz Antonio Buschetto Macarini

SISTEMA DE DETECÇÃO DE DEFEITOS VISUAIS EM PISOS CERÂMICOS BASEADO EM PROCESSAMENTO DE IMAGENS E APRENDIZADO DE MÁQUINA

Este Trabalho de Conclusão de Curso foi julgado aprovado para a obtenção do Título de "Bacharel em Engenharia de Computação" e aprovado em sua forma final pela Universidade Federal de Santa Catarina.

Araranguá, 06 de dezembro de 2017.

Prof ª. Eliane Pozzebon, Drª.
Coordenadora do Curso

Banca Examinadora:

Primeiro membro
Prof. Tiago Oliveira Weber, Dr.
Universidade Federal de Santa Catarina

Segundo membro
Prof. Fabrício de Oliveira Ourique, Dr.
Universidade Federal de Santa Catarina

Terceiro membro
Prof. Marcelo Daniel Berejuck, Dr.
Universidade Federal de Santa Catarina

Este trabalho é dedicado aos meus pais,
a minha família e aos meus amigos.

AGRADECIMENTOS

Eu gostaria de agradecer aos meus pais, Joacir e Arlete, pois eles são a minha base e sem eles eu não teria chegado até aqui. A minha namorada Thayane, pelo suporte e apoio incondicional. A minha irmã, Maria Laura e ao restante da minha família por todo apoio e incentivo durante este período. Aos meus amigos por toda ajuda e pelos momentos que passamos juntos. Gostaria de fazer um agradecimento especial ao meu amigo Luan Casagrande, que me ajudou muito durante a execução deste trabalho. Ao professor e orientador Tiago Oliveira Weber por toda ajuda, conhecimento compartilhado, pela orientação e pela paciência durante o período em que trabalhamos juntos. A Universidade Federal de Santa Catarina pelas oportunidades que me foram dadas e a toda comunidade do campus Araranguá. Por último, mas não menos importante, a Deus, por ter me dados saúde e força durante todo o caminho.

A vida não é fácil. Acostume-se com isso.
(Bill Gates)

RESUMO

A indústria cerâmica possui um sistema de produção altamente industrializado, exceto pelo controle de qualidade. Este ainda é feito por humanos, limitando a velocidade do processo. Os seres humanos podem trabalhar por uma quantidade limitada de horas e seu julgamento é afetado pela fadiga. Sendo assim, o processo poderia ser melhorado utilizando um sistema automatizado para este fim. Neste contexto, o presente trabalho propõe um sistema completo para verificação de defeitos visuais em pisos cerâmicos baseado em processamento de imagens e aprendizado de máquina. O sistema possui quatro etapas: aquisição de imagens, pré-processamento, extração de características e classificação. Na etapa de extração de características, foram comparados dois algoritmos. Na classificação foram testados cinco classificadores, visando buscar o melhor resultado para esta aplicação. O sistema foi implementado utilizando as bibliotecas OpenCV. O sistema apresentou resultados satisfatórios em relação ao tempo de processamento e a taxa de acerto, demonstrando a viabilidade técnica desta proposta.

Palavras-chave: Processamento de Imagens. Aprendizado de Máquina. Visão de Máquina.

ABSTRACT

The ceramic industry has a highly automated production system. The quality control, however, is still performed by humans, which limits its speed. Humans can work during a limited number of hours and get tired, having their judgment affected by the fatigue. This process can be improved using an automated system. In this context, this work proposes a complete verification system for ceramic tiles based on image processing and machine learning. The system has four steps: image acquisition, pre-processing, feature extraction and classification. In the feature extraction step, two algorithms were compared. For classification, five algorithms were tested, aiming to obtain the best result for this application. The system was implemented using OpenCV libraries. The system presented satisfactory results both in processing time and accuracy, showing the viability of the proposed approach.

Keywords: Image Processing. Machine Learning. Computer Vision.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

IA	Inteligência Artificial
AM	Aprendizado de Máquina
Pixel	*Picture Element*
MATLAB	*MATrix LABoratory*
RBF	*Radial Basis Function*
PNN	*Probabilistic Neural Network*
GLCM	*Gray-level Co-Occurence Matrix*
SFTA	*Segmentation-based Fractal Texture Analysis*
CUDA	*Compute Unified Device Architecture*

SUMÁRIO

1 INTRODUÇÃO

A visão é o mais avançado dos nossos sentidos e pode-se dizer que as imagens exercem um dos papéis mais importantes na percepção humana. No entanto, os seres humanos estão limitados à banda visual do espectro eletromagnético. Já os aparelhos de processamento de imagem cobrem praticamente todo o espectro e podem trabalhar com imagens provenientes de várias fontes. Sendo assim, essa área possui um amplo e variado campo de aplicações (GONZALEZ; WOODS, 2010).

A Inteligência Artificial (IA) caminha junto com a área de Processamento de Imagens possibilitando o desenvolvimento de sistemas de visão computacional (GONZALEZ; WOODS, 2010). Mais especificamente, o Aprendizado de Máquina (AM) desempenha um papel fundamental na classificação de padrões, utilizado para verificação de defeitos em diferentes tipos de produtos (FACELI et al., 2011).

A Inteligência Artificial é uma das ciências mais recentes. As pesquisas nesta área começaram logo após a Segunda Guerra Mundial e o próprio nome foi dado em 1956. Esta área não tenta apenas *compreender* as entidades inteligentes. Ela vai muito além, tentando *construí-las* (RUSSELL; NORVIG, 2004).

Em muitos casos, técnicas de IA são utilizadas para solucionar problemas relativamente simples ou até aqueles que são complexos e fazem parte de sistemas maiores. Sendo assim, Coppin (2013) afirma que a "Inteligência Artificial envolve utilizar métodos baseados no comportamento inteligente de humanos e outros animais para solucionar problemas complexos".

Há alguns anos, a Inteligência Artificial era vista como uma área teórica, utilizada apenas para resolver alguns problemas curiosos, mas com pouco valor prático. Estes eram resolvidos por algum algoritmo especificamente codificado para a situação. O conhecimento necessário para a resolução deste tipo de problema era adquirido via entrevistas com especialistas de uma determinada área para descobrir quais regras eram utilizadas nas tomadas de decisão. O *software* era então codificado e à estes programas davam o nome de Sistemas Especialistas ou Sistemas Baseados em Conhecimento (FACELI et al., 2011).

Faceli et al. (2011) ainda afirmam que, nos últimos anos, com a crescente complexidade dos problemas a serem tratados e a quantidade de dados que são geradas por diversas áreas, surgiu a necessidade de criar sistemas mais sofisticados. Isso pode ser feito aumentando sua autonomia, reduzindo a necessidade de intervenção humana e a de-

pendência de especialistas. As técnicas deveriam ser capazes de criar por si próprias, de acordo com experiências passadas, uma hipótese ou uma função, capaz de resolver o problema que se deseja tratar. A capacidade de aprendizado é considerada essencial para um comportamento inteligente. É dado o nome de Aprendizado de Máquina ao processo de indução de uma hipótese (ou aproximação de função) por meio de uma experiência passada.

Mitchell (1997) afirma que o Aprendizado de Máquina é uma área multidisciplinar, que envolve IA, Probabilidade e Estatística, Filosofia, Psicologia, entre outras. O autor ainda define AM como "a capacidade de melhorar o desempenho na realização de alguma tarefa por meio da experiência".

Gonzalez e Woods (2010) definem Processamento Digital de Imagens como "processamento de imagens digitais por um computador digital". À partir da década de 1960, esta área cresceu rapidamente. Atualmente, tais técnicas são utilizadas em uma variedade de aplicações.

Além da utilização na resolução de problemas relativos à percepção de máquinas, a principal área de aplicação destas técnicas é aquela onde os resultados são destinados à interpretação humana. Por exemplo, técnicas utilizadas para restauração de imagens degradadas, previsão do tempo e realce de contraste em radiografias, além de sua aplicação na indústria (GONZALEZ; WOODS, 2010).

Levando em conta os fatos supracitados, este trabalho propõe uma abordagem para a verificação de defeitos visuais em pisos cerâmicos, utilizando processamento de imagens e aprendizado de máquina (*Machine Learning*). O sistema pode ser dividido em quatro etapas: aquisição de imagens, pré-processamento, extração de características (em inglês, *features*) e classificação.

Na primeira etapa, é capturada uma imagem do piso que está na linha de produção. Para simular o ambiente necessário, construiu-se um protótipo de baixo custo, com o objetivo criar um ambiente estável, com o mínimo de influência externa.

Na segunda parte, tem-se como entrada a imagem do piso capturada na etapa anterior. Neste momento, é necessário isolar a região de interesse, removendo o fundo (*background*) da imagem. Feito isto, pode-se corrigir a angulação do piso e qualquer outro problema que possa vir a ocorrer.

Na parte de extração de características, são utilizados algoritmos de processamento de imagem com o intuito de extrair as características que possam descrever o piso em análise. Estas, serão submetidas à quarta etapa do sistema. Nesta, o piso será classificado em *bom* (sem a

presença de defeitos) ou *ruim* (piso com avarias) por um classificador previamente treinado. Visando obter o melhor resultado possível, foram testados e comparados alguns algoritmos de classificação.

1.1 ESTADO DA ARTE

A área de controle de qualidade envolvendo pisos cerâmicos possui uma variedade de trabalhos realizados das maneiras mais diversificadas. Sendo assim, vários métodos e abordagens puderam ser testadas. Optou-se por limitar a pesquisa por trabalhos em cinco anos, trazendo aqueles que foram realizados do ano de 2013 até os dias atuais.

Meena e Mittal (2013) propuseram um método focado em detectar dois tipos de defeitos em pisos cerâmicos: manchas e quebras (ou rachaduras). É baseado em processamento de imagens e operações morfológicas. O objetivo é propor um método que tenha baixo tempo computacional e alta taxa de acerto. O algoritmo possui as fases: aquisição de imagens, aguçamento de contraste, remoção de ruídos, detecção de bordas (este método é dividido em quatro partes), segmentação e operações morfológicas. Na etapa de detecção de bordas, foi utilizado o *Sobel Edge Detector*. O método proposto obteve melhores resultados do que os vistos em Islam e Sahriar (2012), tanto em relação a taxa de acerto quanto ao tempo computacional. Meena e Mittal (2013) obtiveram a média de 97% de acerto. Os defeitos não foram categorizados.

Em Karimi e Asemani (2014), foram propostos algoritmos para detecção de defeitos em pisos. Os autores dividem os métodos de extração de *features* em quatro categorias: *Filtering Methods*, *Structural Algorithms*, *Model Based Techniques* e *Statistical Methods*. A partir disto, são testadas 14 abordagens diferentes dentro destas quatro categorias. O sistema leva em conta os defeitos: *pinhole*, acúmulo de esmalte, rachaduras, manchas, arranhões e problemas de bordas. Alguns métodos mostraram alta velocidade, porém baixa acurácia. Outros, o contrário. Métodos baseados em histograma se destacaram por ter alta velocidade. Além disso, a utilização dos mesmos independe de resolução, o que os tornam úteis para utilização em aplicações de tempo real. A fase de treinamento se mostrou um problema nos métodos baseados em redes neurais. Porém, na fase de testes, houve um desempenho satisfatório se comparado aos outros métodos utilizados no trabalho. Técnicas de morfologia e *Gabor Filters* se mostraram bons candidatos. Após estes fatos, concluiu-se que não existe uma solução geral, já

que estas dependem do tipo de aplicação. Além disso, nenhuma das abordagens obteve um bom desempenho em todos os quesitos.

Em Mishra e Shukla (2014), propõe-se o uso de uma técnica baseada em redes neurais para detectar defeitos em pisos cerâmicos. Foi utilizada uma Rede Neural Probabilistica. O algoritmo é relativamente complexo e utiliza, dentre outras, as técnicas de *edge detection* conhecidas como *Sobel Edge Detector* e *Roberts Detector*, além do *Median Filter*. Depois da aplicação do algoritmo, organiza-se o vetor de características colocando-as em uma única linha, juntamente com a classe correspondente ao defeito que ela possui. A PNN é treinada e utilizada para classificar os defeitos. Entre os detectados pelo sistema, estão: rachaduras, manchas, defeitos de borda e cantos. Todos os resultados obtidos foram melhores do que os vistos em Islam e Sahriar (2012). Obteve-se uma média de 98,18% de acerto utilizando diferentes quantidades de pisos, aumentando em 5% se comparado aos algoritmos anteriores. Além disso, a média do tempo computacional foi diminuído pela metade.

Em Mohan e Kumar (2015), os autores apresentam um sistema automático para detecção de rachaduras em pisos, analisando as imagens capturadas dos mesmos. Serão utilizados métodos de análise de textura. Para classificação, utilizou-se uma PNN (*Probabilistic Neural Network*). As fases do algoritmo são: pré-processamento, segmentação e filtragem morfológica, o que torna o sistema mais flexível e com maior acurácia. Utilizou-se *Discrete Wavelet Transform* e uma *Co-Occurence Matrix*, onde as características extraídas por este método são: energia, entropia, contraste, *correlation coefficience*, e homogeneidade. Para classificação, foi utilizada uma PNN com uma *radial basis function* como função de ativação. Na parte de processamento morfológico, foram utilizadas operações de dilatação e erosão. O trabalho alcançou uma taxa de 98,18% de acerto e comparado com Islam e Sahriar (2012), a taxa de detecção foi melhorada em mais de 5%.

Em Jacob, Shenbagavalli e Karthika (2016), foi proposto um sistema automatizado de detecção de defeitos na superfície de pisos, baseado nas operações de dilatação, erosão, SMEE (*Simple Morphological Edge Extraction*) e técnicas de *Edge Detection*. Utiliza-se um sistema de contagem de pixeis (logo, não há um classificador propriamente dito) para classificar o piso em duas classes: com ou sem defeitos. Se o piso que estiver sendo analisado possuir uma quantidade maior de pixeis do que a imagem de referência, é classificada como defeituosa. Para a obtenção dos resultados, foram utilizados 200 pisos.

Em Hocenski, Matić e Vidović (2016), foi mostrado um protótipo

de estação de visão computacional (*computer vision station*) para verificação em tempo real de *biscoitos*[1] para pisos. O protótipo foi implementado em uma linha de produção de uma cerâmica e procura por defeitos nas bordas, nos cantos e na superfície (manchas, quebras e anormalidades na textura). O tamanho máximo de pisos é 45 cm x 50 cm, com uma velocidade máxima de 0,38 m/s, além de uma distância mínima de 10 cm entre cada piso. Dependendo do tamanho do biscoito, a taxa máxima pode chegar a 60 pisos/minuto. O tempo máximo para a verificação de defeitos foi 1000ms. Alguns parâmetros podem ser ajustados pelo gerente da linha de produção. Foi desenvolvido em C++ utilizando OpenCV e as bibliotecas CUDA. Passa pelos estágios de aquisição, correção de distorção, *thresholding*, detecção de defeitos na borda, nos cantos e na superfície. Por fim, os resultados são mostrados. Os pisos são classificados em bons ou ruins. Os testes foram feitos com 520 pisos brancos. O protótipo procurou por defeitos, assim como a equipe da fábrica. Os resultados são comparados e 513 pisos foram corretamente classificados pela máquina. Houve um acerto de 98,65%.

Em Hanzaei, Afshar e Barazandeh (2017), foi proposto um sistema para detecção e classificação de defeitos em pisos utilizando o operador *Rotation Invariant Measure of Local Variance* (RIMLV) para detecção de problemas na borda, juntamente com uma operação morfológica de fechamento para preencher e suavizar as regiões detectadas. Após isto, todos os defeitos no piso são identificados e as *features* são extraídas. Por último, um *Multi-Class Support Vector Machine* com a estratégia *winner-takes-all* foi utilizado para classificar o tipo de defeito. Utilizando o *Median Filter*, RIMLV e as operações morfológicas, chegou-se a uma média de 93,4% de acerto. A média do tempo de detecção foi de 3,52 segundos.

1.2 JUSTIFICATIVA E MOTIVAÇÃO

A cerâmica é um dos materiais mais antigos da humanidade. Seu nome vem do grego, *keramikós* e significa "argila queimada". Segundo Reed (1995), ela é utilizada desde 5000 a.C. É a partir dela que são produzidos os pisos cerâmicos.

Em 2013, a Ásia, continente líder na produção de pisos, produziu 8315 milhões de metros quadrados, totalizando 69,8% da produção mundial. Em relação aos países, o Brasil, segundo maior produtor do

[1] A palavra "biscoito" se refere ao material bruto do piso, uma etapa antes do mesmo ganhar coloração.

mundo, deteve 7.3% da produção mundial, com 871 milhões de metros quadrados. Em primeiro lugar, encontra-se a China, com uma produção de 5700 milhões de metros quadrados, equivalente à 47,8% da produção mundial (STOCK, 2011).

Segundo o mesmo autor, em relação ao consumo, a Ásia é o continente que mais consome, com 7692 milhões de metros quadrados (66,5% da produção mundial). O Brasil, ocupa novamente o segundo lugar quando se trata de consumo. Em 2013, consumiu 837 milhões de metros quadrados, estes sendo 7,2% da produção de todo o mundo. A China, se encontra em primeiro lugar no *ranking*, com 39,4% do consumo mundial, cerca de 4556 milhões de metros quadrados.

Segundo Gomes et al. (2014), o setor de Cerâmica emprega 19 mil pessoas em Santa Catarina. Esse montante representa 11% dos trabalhadores do setor em todo o Brasil. Entre estes empregos, a maior parte se concentra na região da Grande Florianópolis e na Região Sul, tendo como principais municípios: Criciúma, Tijucas e Sangão, onde há 8 mil empregos, este valor equivalendo à 42% dos trabalhadores do setor no território estadual.

Gomes et al. (2014) ainda afirma que o setor ceramista produziu R$ 2,16 bilhões em 2011, o dobro do segundo colocado, o setor de Concreto. Já em relação ao comércio exterior, o setor Cerâmico movimentou, em 2012, cerca de US$ 166 milhões. Já nas importações, foram movimentados pouco menos de US$ 111 milhões no mesmo ano.

A produção de cerâmica é um processo industrializado. Porém, ainda é necessário que haja intervenção humana em cada peça. O controle de qualidade é o processo que mais depende desta intervenção. É um procedimento difícil, intensivo e que ocorre em um ambiente industrial pesado, como muito barulho, alta temperatura e umidade. Segundo Elbehiery, Hefnawy e Elewa (2005), este processo pode ser dividido em: análise de cor, verificação de dimensões e de defeitos visuais. Estes, podem se originar de impurezas químicas ou problemas físicos durante o processo.

Além disso, a inspeção de pisos cerâmicos requer pessoas especializadas. O problema disto, é que estas podem ter opiniões diferentes sobre a presença de defeitos. A capacidade humana depende de treinamento, conhecimento e experiência (MEENA; MITTAL, 2013).

Segundo o mesmo autor, o ponto negativo deste processo ser feito por humanos, é que o mesmo pode trabalhar por um tempo limitado. Ou seja, fica facilmente cansado em algumas horas. Assim, o seu julgamento é afetado pela fadiga. Outro fator negativo, é que a taxa de saída dos pisos da parte de controle de qualidade não condiz com a

taxa de produção dos mesmos. Ou seja, há um "gargalo" neste ponto.

1.2.1 Definição do Problema

Apesar de ser uma área com uma quantidade relativamente grande de trabalhos, cada um dos que foram citados na Seção 1.1 possuem uma limitação em pelo menos algum quesito.

Um sistema de detecção de problemas em pisos na indústria cerâmica precisa funcionar em tempo real. Ou seja, necessita de um tempo de processamento muito baixo. Isso se dá pelo fato de que a taxa de produção é extremamente alta. No trabalho de Hanzaei, Afshar e Barazandeh (2017), a média do tempo para a detecção de defeitos foi de 3,52 segundos e é considerado inaceitável para os padrões industriais.

A alta taxa de acerto na classificação dos pisos é outro requisito imprescindível na implementação de um sistema de controle de qualidade. Quando o contrário acontece, a empresa perde dinheiro e a confiança dos clientes, já que a qualidade dos produtos que saem da fábrica não é boa. No trabalho de Hanzaei, Afshar e Barazandeh (2017), a taxa média de acerto foi 93,4%. Apesar de ser relativamente alta, existe a possibilidade de ela ser aumentada.

Como a intenção é apresentar uma abordagem que possa ser implementada em uma linha de produção, existe a necessidade de que esta possa classificar pisos de qualquer tamanho. No sistema proposto por Hocenski, Matić e Vidović (2016), este precisa ter as dimensões máximas de 45 cm $\times$ 50 cm. Em determinadas empresas cerâmicas são fabricados pisos que possuem dimensões maiores que estas. Nestes casos, a verificação de defeitos teria que novamente ser feita pelos humanos.

Citando novamente o fato de estar sendo proposta uma abordagem que possa ser aplicada na indústria, a necessidade de que esta encontre defeitos em pisos com diferentes tipos de texturas e cores se torna necessário. No trabalho de Hocenski, Matić e Vidović (2016), os resultados foram obtidos apenas com pisos brancos. Os autores afirmam que, em trabalhos futuros, espera-se utilizar diferentes tipos, inclusive com texturas.

Segundo Elbehiery, Hefnawy e Elewa (2005), na indústria cerâmica os defeitos são originários de problemas físicos ou impurezas químicas e podem causar inúmeras consequências. No trabalho de Meena e Mittal (2013) e Mohan e Kumar (2015), apesar de ambos possuírem uma alta taxa de acerto, são detectados apenas dois tipos de defeitos no primeiro

trabalho e quatro no segundo. Isso faz com que o sistema não possa ser implementado em uma linha de produção, já que nesta existe uma variedade maior de defeitos.

Todos os problemas citados anteriormente acontecem de modo prático na indústria. No trabalho de Mishra e Shukla (2014), mesmo havendo bons resultados, eles foram obtidos através de simulação. Apesar de serem válidos, podem não refletir o que acontece nas fábricas e consequentemente, não resolve os problemas contidos no processo.

Por todos estes motivos, se faz necessário um sistema que automaticamente classifique os pisos e demande um baixo tempo computacional. Além disso, é necessário haver alta taxa de acerto e que funcione para uma variedade de pisos, com ou sem textura. Este classificará todas as placas de acordo com o mesmo critério e não sofrerá de fadiga.

1.3 OBJETIVOS

1.3.1 Objetivo Geral

Demonstrar que a abordagem para a detecção de defeitos visuais em pisos cerâmicos proposta neste trabalho tem potencial para ser aplicada na indústria.

1.3.2 Objetivos Específicos

1. Utilizar técnicas de processamento de imagem e métodos de análise de textura para obter um tempo de processamento baixo;

2. Fazer o uso de técnicas de aprendizado de máquina tendo como objetivo atingir uma taxa de acerto satisfatória se comparada com a literatura;

3. Demonstrar que a abordagem proposta pode classificar pisos de qualquer tamanho;

4. Pesquisar e implementar técnicas de processamento de imagem para que a abordagem possa detectar a maior quantidade de defeitos possíveis;

5. Usar técnicas da mesma área com o intuito de classificar a maior

variedade de pisos possíveis, de cores diferentes, com ou sem texturas.

1.4 METODOLOGIA

Como metodologia geral deste trabalho, pretende-se propor, desenvolver, testar e validar um novo método para classificação de pisos utilizando processamento de imagens e aprendizado de máquina. Mais especificamente:

1. Levantar o estado da arte no que se diz respeito à controle de qualidade de pisos envolvendo processamento de imagens;
2. Desenvolver uma abordagem capaz de classificar visualmente pisos cerâmicos, de acordo com a presença ou não de defeitos;
3. Implementar vários classificadores para uma posterior comparação de resultados;
4. Integrar o sistema descrito em (2) com os classificadores descritos em (3) e descobrir a melhor combinação (aquela que traz os melhores resultados);
5. Testar e validar o sistema integrado descrito no item anterior.

2 VISÃO COMPUTACIONAL

Neste capítulo serão apresentados alguns conceitos sobre Visão Computacional e Processamento de Imagens, já que estes são necessários para compreender o funcionamento do sistema que está sendo proposto. Inicialmente, serão apresentados os conceitos básicos inerentes à esta área. Depois, serão abordados aqueles que possuem um pouco mais de complexidade e que, eventualmente, serão utilizados para o desenvolvimento do sistema proposto.

Desde o início da ciência, a observação sempre teve um papel muito importante. Antigamente, a única maneira de documentar os resultados era a descrição verbal e os desenhos manuais. A invenção da fotografia foi o evento que mudou isso. Um grande exemplo são os astrônomos. Estes foram capazes de medir as posições e os tamanhos das estrelas. O problema é que estes procedimentos demandavam uma grande quantidade de tempo (JÄHNE, 2002).

Jähne (2002) ainda afirma que estamos no meio de uma *segunda revolução*, onde o progresso da tecnologia envolvendo computação está crescendo rapidamente. Agora os computadores pessoais e *workstations* tem poder suficiente para processar imagens. Assim, são capazes de lidar com processamento de sequências de imagens e até modelos 3D.

A área de visão computacional foi chave para o desenvolvimento dessa tecnologia. Marr (1982) afirma que um sistema de Visão Computacional é aquele que executa a mesma tarefa que a visão humana para descobrir o que está presente no mundo e onde é que estas coisas se localizam. O objetivo geral da Visão Computacional é obter informações sobre uma cena através da análise computacional. Isso acontece de acordo com as entradas recebidas. Estas podem ser uma ou mais imagens digitais (BASU; LI, 1993). Em contraponto, o termo *visão de máquina* é utilizado para sistemas que executam tarefas como checar tamanho e integridade de peças no ambiente industrial, por exemplo (JÄHNE, 2002).

2.1 PROCESSAMENTO DIGITAL DE IMAGENS

A visão humana é um dos mecanismos mais complexos e importantes, pois entrega dados ao cérebro. Ela provê informações para tarefas relativamente simples, como o reconhecimento de um objeto e para tarefas mais complexas, como uma tomada de decisões (PITAS,

2000).

Pitas (2000) ainda afirma que os primeiros esforços empregados na àrea de processamento de imagens começaram no *Jet Propulsion Laboratory* (Pasadena, Califórnia), em 1964. As primeiras aplicações foram dedicadas a processar imagens vindas da lua. Assim, um novo ramo da ciência chamado de *processamento digital de imagens* se iniciou. Gonzalez e Woods (2010) afirmam que o processamento digital de imagens se refere ao processamento de imagens digitais por um computador digital.

Ainda é difícil ter uma noção exata do ponto em que o processamento de imagens termina e outras áreas, como a análise de imagens, começam. Algumas vezes, esta é definida como uma disciplina na qual tanto a entrada quanto a saída de um processo são imagens digitais. Mas isso nem sempre é verdade. Um exemplo é o cálculo da intensidade média de uma imagem. Este procedimento resulta em um único número. Pela definição utilizada anteriormente, esta operação não seria considerada como sendo de processamento de imagens (GONZALEZ; WOODS, 2010).

Mesmo que os limites entre as áreas não sejam claros, Gonzalez e Woods (2010) afirmam que é útil levar em conta a diferenciação de três tipos de processo: os de baixo, médio e alto nível. São consideradas de baixo nível aquelas operações mais primitivas, de pré-processamento, como a redução de ruído. Seus processos são caracterizados por ter uma imagem como entrada e saída. Os de nível médio são aqueles que envolvem tarefas como a segmentação (separação da imagem em regiões ou objetos). Como entrada, tem-se uma imagem e como saída, os atributos extraídos. Já os processos de nível alto são caracterizados por tentar "dar sentido" a um conjunto de objetos reconhecidos. Seria como tentar realizar funções cognitivas normalmente associadas à visão.

2.2 IMAGEM DIGITAL

Uma imagem digital pode ser considerada como uma representação discreta de dados possuindo informações espaciais (sobre o formato) e sobre intensidade (SOLOMON; BRECKON, 2011).

Segundo Gonzalez e Woods (2010), uma imagem é uma função bidimensional, $f(x, y)$, em que x e y são coordenadas *espaciais* e a amplitude de f em qualquer posição deste plano é chamada de *intensidade* ou *nível de cinza*. Quando x, y e os valores de nível de cinza de f são valores finitos e discretos, chamamos de *imagem digital*.

Já Basu e Li (1993) afirmam que a imagem digital é a entrada para um computador digital por meio de amostragem do seu brilho em uma matriz. Os elementos desta são chamados de *pixels*.

2.2.1 *Pixel*

A palavra *pixel* é uma abreviação de *picture element*. Também pode ser conhecido como *elementos pictóricos*, *elementos de imagem* ou *pels*. Representam o menor elemento constituinte em uma imagem digital. Contém um valor numérico que representa a unidade básica de informação em uma imagem, com uma determinada resolução e nível de quantização (SOLOMON; BRECKON, 2011).

Solomon e Breckon (2011) ainda afirmam que geralmente os pixeis representam o valor de cor ou intensidade de uma imagem como uma pequena amostra de "luz colorida" da cena. Entretanto, nem todas as imagens contém apenas informações visuais. Uma imagem nada mais é do que um sinal 2D digitalizado como um *grid* de pixeis e estes valores informam outras propriedades além de cor e intensidade de luz. A informação contida em *pixel* pode variar muito de acordo com o tipo de imagem que está sendo processada.

2.2.2 Representação de uma imagem digital

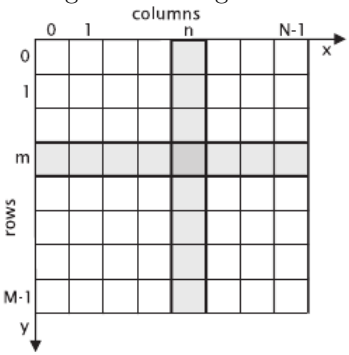
Computadores não conseguem lidar com imagens contínuas, mas apenas vetores numéricos. Por isso, as imagens são representadas por matrizes e a localização dos pixeis é dada de acordo com a notação das mesmas (JÄHNE, 2002).

O primeiro índice, m, denota a posição de uma linha e o segundo, n, denota a posição de uma coluna, como pode ser visto na Figura 1. Se a imagem contém $M \times N$ pixeis, ou seja, é representada por uma matriz $M \times N$, o índice n varia de 0 até $N - 1$ e m, de 0 à $M - 1$. M é a quantidade de linhas e N representa o número de colunas.

2.3 AQUISIÇÃO DE IMAGENS

Um sistema de aquisição de imagens capta a radiação emitida por objetos para torná-los visíveis. Em visão computacional, as cenas e a iluminação são fatores que não requerem tanta atenção quanto num sistema de aquisição. A percepção do objeto pode ser influenciada pela

Figura 1 – Imagem 2D



Extraído de: (JÄHNE, 2002)

maneira com que o sistema está disposto. Por isso, é importante que ele esteja montado de modo correto. Isso fará com que as distorções sejam minimizadas (JÄHNE, 2002).

O processo de aquisição é de suma importância e precisa ser feito de maneira correta. Qualquer deficiência nas imagens adquiridas pode fazer com que muitos problemas apareçam na etapa de análise e interpretação. Um exemplo comum é a falta de detalhes devido a deficiência no contraste ou um ajuste de foco ruim. Isso pode fazer com que a medida de dimensões em um objeto se torne uma tarefa impossível, ou ainda, pode torná-lo irreconhecível (DAVIES, 2012).

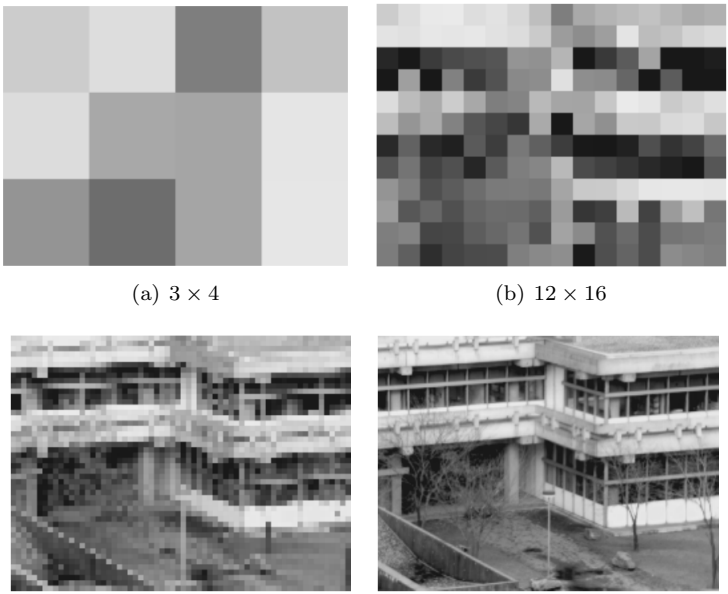
A complexidade do sistema de aquisição depende das operações que serão realizadas na imagem. Por exemplo, se a intenção é fazer a medida das dimensões de um objeto, basta ter um sistema em que o mesmo fique uniformemente iluminado e possa ser claramente distinguido do cenário ao fundo (JÄHNE, 2002).

2.3.1 Amostragem e quantização

É difícil chegar a uma conclusão sobre quantos pixeis são suficientes para representar uma imagem. Não existe uma resposta geral sobre o assunto. Empiricamente, sabe-se que a quantidade deve ser suficiente para que os objetos possam ser distinguidos dentro da imagem. Geralmente, estes valores são limitados pelo sensor da câmera utilizada (JÄHNE, 2002).

Amostragem e *quantização* são dois processos utilizados para

Figura 2 – Representação de uma imagem com diferentes quantidades de pixeis



(a) 3×4 (b) 12×16

(c) 48×64 (d) 192×256

Extraído de: (JÄHNE, 2002)

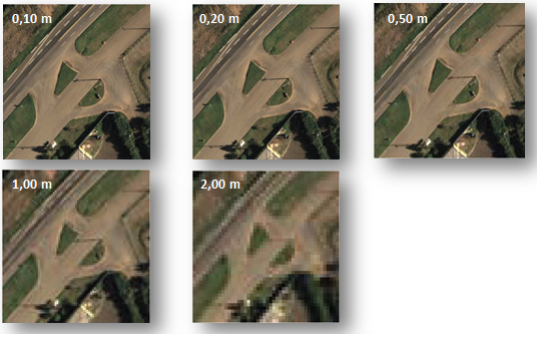
converter os dados contínuos capturados pelo sensor de uma câmera em uma imagem digital. Levando em conta uma imagem f que é contínua em relação as coordenadas x, y e a amplitude, a *amostragem* é o processo de digitalização dos valores das coordenadas. O mesmo procedimento relacionado a amplitude é chamado de *quantização* (GONZALEZ; WOODS, 2010).

2.3.2 *Ground Sample Distance*

O *Ground Sample Distance* (GSD) é a medida de limitação devido a amostragem (HIGGINS; WOLFE, 1955). É a representação do pixel da imagem em uma unidade. Esta medida está relacionada com a altura em que a câmera está posicionada levando em conta o objeto em análise. A escolha do GSD influencia diretamente na nitidez das imagens. Quanto menor é o seu valor, maior é o nível de detalhamento.

Ou seja, o tamanho do GSD é inversamente proporcional ao nível de detalhamento (NETO, 2016). A Figura 3 mostra um exemplo de como o GSD influencia na resolução das imagens.

Figura 3 – A influência do GSD no nível de detalhamento das imagens



Extraído de: (NETO, 2016)

Observando a Figura 3, pode-se ver imagens capturadas de uma determinada área com diferentes valores de GSD. Na primeira, seu valor é de 10 cm e a nitidez é maior. Na última, onde o mesmo tem o valor de 2m, o nível de detalhamento é menor.

A grosso modo, pode-se dizer que o GSD é o valor de tamanho que um *pixel* representa na vida real. Ou seja, quando o seu valor é de 10 cm, pode-se dizer que cada *pixel* na imagem é equivalente a esta medida no ambiente real.

O valor do GSD para determinada aplicação pode ser aproximado de modo teórico. Seu cálculo (Equação 2.1) depende do *sensor width* (S_w) da câmera, do *focal length* da mesma (F_R), da altura em que a câmera está localizada em relação ao objeto em análise (H) e da largura da imagem (imW) (PIX4D, 2017).

$$GSD = \frac{S_w \cdot H \cdot 100}{F_R \cdot imW} \tag{2.1}$$

S_w deve ser dado em milímetros, assim como F_R. Já H deve estar em metros e imW em pixeis. Assim sendo, tem-se como resultado um valor de GSD em centímetros/pixel.

2.4 ESPAÇO DE CORES

Em 1666, Sir Isaac Newton descobriu que, quando um feixe de luz solar atravessa um prisma de vidro, o resultado do outro lado não é branco, mas consiste em um espectro contínuo de cores, variando de violeta à vermelho. Este, pode ser dividido em seis grandes regiões: violeta, azul, verde, amarelo, laranja e vermelho. Nenhuma cor do espectro termina abruptamente, pois cada uma se funde suavemente com a próxima (GONZALEZ; WOODS, 2010).

O objetivo de um espaço de cores (também conhecido como *modelo de cores* ou *sistema de cores*) é especificar as cores de uma forma padronizada, amplamente aceita. Basicamente, é uma especificação de um sistema de coordenadas e um subespaço dentro deste, no qual cada cor é representada por um único ponto. Em termos de processamento de imagens, um dos modelos de cores mais utilizados é o RGB (GONZALEZ; WOODS, 2010).

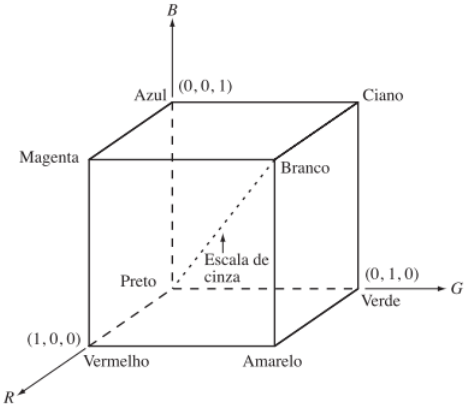
2.4.1 RGB

No modelo de cores RGB (***R**ed*, ***G**reen*, ***B**lue*), cada cor aparece em seus componentes espectrais de vermelho, verde e azul. O modelo se baseia no sistema de coordenadas cartesianas e o subespaço de cores de interesse é o cubo (três planos 2D), no qual os valores RGB primários estão em três vértices. O preto está na origem e o branco no vértice mais distante (Figura 4). A escala de cinza estende-se do preto ao branco pelo segmento de reta que une os dois pontos. Por conveniência, assume-se que os valores de cor foram normalizados. Ou seja, seus valores estão dentro de um intervalo [0, 1] (GONZALEZ; WOODS, 2010).

O modelo RGB é o mais utilizado para representar imagens digitais. Os canais podem ser facilmente separados. É importante lembrar que as cores presentes em uma imagem são a mistura dos componentes de cor dos três canais. Um erro muito comum é achar que, por exemplo, as partes azuis de uma imagem só irão aparecer no canal B (*Blue*). Estes itens irão certamente aparecer de modo mais claro neste canal. Mas deve-se lembrar que mesmo um objeto azul possui uma mistura de componentes de cor vindos dos canais R e G (SOLOMON; BRECKON, 2011).

Solomon e Breckon (2011) ainda afirmam que em processamento digital de imagens, é utilizado um modelo RGB simplificado, otimizado e padronizado para *displays* gráficos, baseado no padrão de cores CIE,

Figura 4 – Espaço de cores RGB representado por um cubo



Extraído de: (GONZALEZ; WOODS, 2010)

de 1931.

A quantidade de *bits* utilizados para representar cada *pixel* no espaço RGB é chamado de *profundidade de pixel*. Utilizando como exemplo uma imagem para qual cada canal seja representado por 8 *bits*, pode-se dizer que cada *pixel* de cor RGB tem uma profundidade de 24 *bits*: três canais com 8 *bits* cada. O termo *full-color* ou *imagem colorida* é utilizado para caracterizar uma imagem RGB de 24 *bits* (GONZALEZ; WOODS, 2010).

2.5 TIPOS DE IMAGENS DIGITAIS

Antes de desenvolver um algoritmo para processamento de imagens, é necessário levantar algumas questões, como por exemplo: quando é necessário utilizar uma imagem colorida ou uma em escala de cinza? A primeira, se torna importante por facilitar na hora de fazer algumas análises. Porém, traz uma necessidade de maior capacidade de armazenamento e é necessário um maior poder de processamento (DAVIES, 2012).

Davies (2012) ainda sugere que em alguns casos, é necessária uma boa discriminação de cores para poder separar (segmentar) objetos em uma imagem. Porém, às vezes, pode-se deixar de utilizar um ou dois canais, diminuindo a quantidade de informações que devem ser processadas. Exceto pelos casos onde o uso de uma imagem colorida

se faz necessária, a escolha da mesma em lugar de uma em escala de cinza deve ser devidamente justificada.

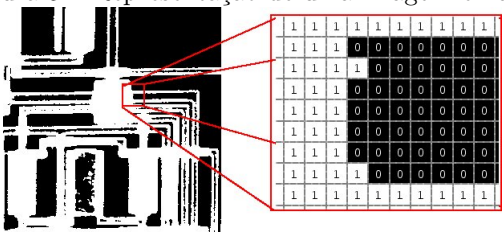
As imagens coloridas são aquelas onde o espectro de cores pode ser representado por um conjunto de três valores, tipicamente os componentes RGB de determinada região. Cada cor é representada por uma combinação linear das cores básicas e a imagem pode ser interpretada como três planos de duas dimensões (SOLOMON; BRECKON, 2011).

Imagens em escala de cinza (*gray scale images*, também conhecidas como *intensity images*) são representadas por uma matriz onde cada elemento possui um valor correspondente à quão claro ou escuro é um *pixel*, de acordo com a sua cor, em determinada posição (SANDBERG, 2000).

São atribuídos valores de preto (zero) até branco (máximo), de acordo com o nível de sinal. Este tipo de escala é muito útil para descrever a intensidade das imagens, pois pode expressá-la como um único valor em cada ponto da imagem (SOLOMON; BRECKON, 2011).

Uma imagem binária é representada por uma matriz onde os pixeis podem ser pretos ou brancos e nenhuma outra coisa além destas. O valor "0" é atribuído à cor preta e "1" à cor branca (SANDBERG, 2000).

Figura 5 – Representação de uma imagem binária



Extraído de: (MATHWORKS, 2017)

2.5.1 Conversão de RGB para *Grayscale*

Esta conversão pode ser feita utilizando uma transformada simples. Processar uma imagem em escala de cinza é melhor, computacionalmente falando, pois ela contém uma quantidade relativamente menor de informações. Por isso, se torna o primeiro passo em uma série de algoritmos de processamento de imagens. As informações importan-

tes da imagem são mantidas, como por exemplo, as bordas, regiões, manchas, entre outras (SOLOMON; BRECKON, 2011). Analisando a Figura 6, pode-se verificar que todos os objetos da imagem podem ser claramente distinguidos.

Figura 6 – Diferença entre uma imagem em RGB e *Grayscale*

(a) Imagem em RGB (b) Imagem em *Grayscale*

Adaptado de: (ROSENBERG, 2001)

Solomon e Breckon (2011) afirmam que uma imagem RGB (I_{RGB}) pode ser convertida para uma imagem em escala de cinza ($I_{Grayscale}$) utilizando a seguinte equação:

$$I_{Grayscale}(n,m) = \alpha I_{RGB}(n,m,r) + \beta I_{RGB}(n,m,g) + \gamma I_{RGB}(n,m,b) \tag{2.2}$$

Onde (n,m) são os índices de cada *pixel* na imagem em *grayscale* e (n,m,c) são os valores dos *pixels* presentes em cada canal c da imagem RGB. Esta conversão é irreversível. Os valores de cor presentes na imagem colorida não podem ser recuperados no caso de uma conversão inversa.

O mesmo autor afirma que pela Equação 2.2, pode-se perceber que a imagem em escala de cinza é formada por uma soma ponderada dos canais *red*, *green* e *blue*. Os coeficientes (α, β e γ) são ajustados de acordo com a proporção da resposta visual do olho humano aos canais supracitados. Além disso, eles asseguram a uniformidade da imagem.

2.6 SEGMENTAÇÃO

Dá-se o nome de segmentação ao processo de subdividir imagens em suas regiões constituintes ou objetos. Tem um papel muito importante porque geralmente é um dos primeiros passos em um algoritmo de processamento de imagens. Por isso, precisa ser feito de modo correto, já que os processos subsequentes dependem dele. Se não é possível identificar um objeto, não existe a possibilidade de classificá-lo ou descrevê-lo (SOLOMON; BRECKON, 2011).

O objetivo básico da segmentação é separar a imagem em partes mutuamente exclusivas, onde estas podem ser devidamente identificadas. A parte extraída no processo pode ser chamada de *foreground* e o restante da imagem é o *background*. Não é certo afirmar que existe apenas um modo correto de segmentação, já que isto depende muito do que se quer identificar na imagem (SOLOMON; BRECKON, 2011).

Solomon e Breckon (2011) afirmam que a questão central em segmentação de imagens é escolher o tipo de relacionamento entre pixeis que deve ser levado em conta na hora de decidir a qual região este pertence. Existem basicamente dois tipos de abordagens:

- Métodos de borda: baseia-se na detecção de bordas, tendo como objetivo identificar os limites entre as regiões. Na maioria dos casos, é necessário encontrar as diferenças abruptas de intensidade entre os grupos de pixeis;

- Métodos baseados em regiões: classifica-se os pixeis de acordo com o seu nível de semelhança.

2.6.1 Uso das características da imagem para segmentação

Para que a segmentação seja feita de modo correto, muitas vezes avaliar somente a intensidade dos pixeis não basta. É necessário utilizar propriedades e características mais sofisticadas. As três mais utilizadas são:

- Cor: em casos onde um objeto possui uma cor muito diferente do *background*, é simples separá-los. Por exemplo, retirar uma laranja que está colocada sobre uma toalha azul;

- Textura: apesar de ser um conceito relativo e não possuir uma definição absoluta, pode ser geralmente definida como *a variação*

dos valores de cor ou intensidade em um determinado espaço. A maneira de medir isso depende de valores estatísticos, como a variância ou a intensidade em uma certa vizinhança;

- *Motion*: uma sequência de *frames* (quadros) pode ser utilizada como uma maneira de segmentação. Quando existe um *background* estático, uma operação de subtração é suficiente para retirar objetos de uma imagem, ainda que estes estejam se movendo.

Além dos itens citados anteriormente, pode-se realizar uma combinação entre eles e tornar o processo de segmentação mais preciso (SOLOMON; BRECKON, 2011).

2.7 *THRESHOLDING*

Muitas vezes chamado de *Intensity Thresholding* ou *Binary Thresholding*, é o método de segmentação mais simples. Escolhe-se um valor de limiar (do inglês, *threshold*) e os pixeis que possuírem o valor de intensidade maior do que este são designados a uma determinada região. Caso contrário, são colocados em outra. Pode ser matematicamente definido como:

$$b(x,y) = \begin{cases} 1, & \text{se } I(x,y) > T \\ 0, & \text{caso contrário.} \end{cases} \tag{2.3}$$

Onde $b(x,y)$ é a imagem binária gerada após a operação de *thresholding* e T é o valor do limiar. Em casos simples, é fácil chegar a um resultado satisfatório.

A escolha manual do limiar é feita por tentativa e erro. A este tipo de abordagem dá-se o nome de *global thresholding*. Porém, o grande problema deste método é que provavelmente não funcionará em todos os casos, se fosse necessário utilizar em uma aplicação real. Por isso, nem sempre é aceitável utilizá-lo (SOLOMON; BRECKON, 2011).

Solomon e Breckon (2011) ainda afirmam que a seleção automática de limiar se baseia na consideração do histograma da imagem. Quando é possível fazer este tipo de operação, ao plotar-se o gráfico, fica claro dois picos no mesmo: um referente ao objeto e outro ao *background*. Assim, fica mais simples de encontrar um limiar que possa segmentar a imagem de modo correto.

2.8 *K-MEANS CLUSTERING*

O *k-Means Clustering* é um algoritmo de aprendizado não-supervisionado utilizado quando há dados sem identificação. O objetivo deste é encontrar grupos (*clusters*) para os dados. A quantidade de *clusters* é definida pela variável k (TREVINO, 2016).

O algoritmo consiste em um processo iterativo e pode ser dividido basicamente em três etapas. O primeiro passo consiste em escolher duas centroides (ou mais, dependendo da aplicação) aleatoriamente ou utilizando um algoritmo de inicialização de centro (*center initialization algorithm*). Então, a distância entre cada ponto e as centroides são calculadas. Esta, é calculada de acordo com a similaridade dos dados com a centroide. Os dados são agrupados de acordo com aquela que está mais próxima (OpenCV, 2014).

O último passo consiste em calcular a média de distância entre as amostras e suas respectivas centroides. Baseado neste resultado, as novas posições destas são encontradas. O segundo e o terceiro passo são repetidos até que o critério de parada seja atingido. Este pode ser a quantidade de iterações ou quando a acurácia determinada foi alcançada, por exemplo (OpenCV, 2014).

2.9 TRANSFORMADA *AFFINE*

A transformada *Affine* pode ser considerada como qualquer transformada que pode ser expressa na forma de multiplicação de matrizes (transformação linear) seguida por uma soma de vetores (translação). De maneira resumida, a transformada *Affine* representa a relação entre duas imagens. Sendo assim, pode ser utilizada para expressar: rotações (transformação linear), translação (adição de vetores) ou operações de mudança de escala (transformação linear) (OpenCV, 2015).

É uma técnica geralmente utilizada para corrigir distorções geométricas ou deformações (MATLAB, 2017). Como citado anteriormente, as principais operações de transformação 2D são a translação, a rotação e a mudança de escala. A utilização de coordenadas e transformadas homogêneas garantem que as operações mencionadas possam ser representadas em forma de matriz (BISWAS et al., 1996).

2.10 *EDGE DETECTION*

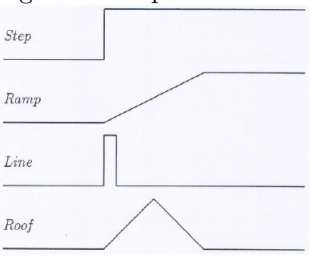
As técnicas de *edge detection* estão entre as principais operações na área de processamento de imagens. Por isso, é necessário ter domínio sobre estes tipos de algoritmos. Elas estão entre as operações mais utilizadas em análise de imagem e são muito usadas para fazer detecção de objetos (SHARIFI; FATHY; MAHMOUDI, 2002). A utilização destes detectores geralmente faz o processo de segmentação e reconhecimento de padrões se tornar mais simples. Além disso, reduzem a quantidade de informações que precisam ser processadas, filtrando aquelas que são desnecessárias (MALIK; KUMAR, 2016).

Sharifi, Fathy e Mahmoudi (2002) afirmam que a borda em uma imagem é definida como uma descontinuidade nos valores de nível de cinza. Isso acontece porque existe uma variação muito grande na intensidade dos pixeis e assim, as bordas do objeto se tornam visíveis. Na aplicação de filtros 2D, assume-se que os pixeis que formam as bordas dos objetos são aqueles que possuem um alto gradiente (SHRIVAKSHAN; CHANDRASEKAR, 2012).

Estas descontinuidades presentes nas imagens podem ser do tipo degrau (do inglês, *step*), onde as mudanças são abruptas. Além disso, existem aquelas conhecidas como *line*, onde a intensidade muda rapidamente e depois de um curto espaço de tempo, ela volta ao valor antigo (JAIN; KASTURI; SCHUNCK, 1995).

Jain, Kasturi e Schunck (1995) ainda afirmam que apesar de existirem os tipos de borda citados anteriormente, os mesmos são raros em uma imagem real. Geralmente, as bordas do tipo degrau se tornam rampas (do inglês, *ramps*) e as do tipo "linha" se tornam *roof*, onde a mudança de intensidade não acontece instantaneamente. Estes tipos de borda podem ser vistos na Figura 7.

Figura 7 – Tipos de bordas

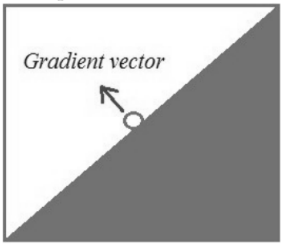


Extraído de: (JAIN; KASTURI; SCHUNCK, 1995)

A Figura 8 mostra como seria, idealmente, um *pixel* de borda e o seu gradiente. Supondo uma imagem em escala de cinza, percebe-se que a intensidade varia de 0 a 255 na direção do gradiente. A magnitude do mesmo indica quão brusca é essa variação (SHRIVAKSHAN; CHANDRASEKAR, 2012).

Shrivakshan e Chandrasekar (2012) afirmam que em imagens reais, geralmente não existem bordas ideais. É por isso que se usa um valor de limiar para a detecção das mesmas: se a magnitude do gradiente for maior que este valor, o ponto é corresponde à um *pixel* de borda.

Figura 8 – Um *pixel* de borda e seu gradiente



Extraído de: (SHRIVAKSHAN; CHANDRASEKAR, 2012)

Um *pixel* de borda pode ser descrito por duas características. A primeira, é a *edge strength*. Esta é definida como a magnitude do gradiente (Equação 2.4). A segunda, é a direção da borda, chamada de *edge direction*. Ela corresponde ao ângulo do gradiente (Equação 2.5).

$$|\vec{grad}| = \sqrt{\left(\frac{\partial}{\partial x}\right)^2 + \left(\frac{\partial}{\partial y}\right)^2} \tag{2.4}$$

$$\Psi = \arctan\left(\frac{\partial}{\partial y} / \frac{\partial}{\partial x}\right) \tag{2.5}$$

Onde x e y são as coordenadas dos pixeis, $\vec{grad}$ é a magnitude do gradiente e Ψ é o ângulo do mesmo (SHRIVAKSHAN; CHANDRASEKAR, 2012).

A direção do gradiente é perpendicular à orientação da borda. Em muitos algoritmos, a direção é utilizada para localizá-las. O módulo do gradiente é não-linear, invariante à rotação e é calculado utilizando as derivadas em x e y. Em imagens ruidosas, pode ser útil usar várias derivadas direcionais para aumentar o *signal-to-noise ratio* (ZIOU; TAB-

BONE et al., 1998).

Em uma imagem, separar um objeto do seu fundo é um passo essencial para a sua interpretação. É uma operação simples para o sistema visual humano. Mas quando um algoritmo tem de realizar uma tarefa como esta, alguns problemas podem ser encontrados. Por exemplo, quando há muito ruído ou problemas na hora de quantizar a imagem, podem ser detectadas bordas onde não existem. Além disso, pode ocorrer o processo inverso: não conseguir detectar bordas que efetivamente existem (BASU, 2002).

Basu (2002) ainda cita como um possível problema o fato de a imagem ter uma quantidade de ruído muito alta. Isso pode fazer com que a borda localizada esteja deslocada de sua posição real. Além disso, pode-se encontrar imagens onde a variação de intensidade da borda se dá por componentes de alta frequência e não necessariamente pelas bordas em si. Por isso, na hora de aplicar um filtro para suavizar a imagem, a verdadeira borda vai desaparecer, tornando-a impossível de ser encontrada.

Jain, Kasturi e Schunck (1995) afirmam que a maioria dos algoritmos de detecção de borda contém três passos. São eles:

1. Filtragem: O cálculo do gradiente baseado na intensidade de dois pontos é sujeito a ruído. A utilização de filtros tem como principal objetivo aumentar a performance do algoritmo no que diz respeito a este fato. O problema disso é que a *edge strength* das bordas podem diminuir quando uma técnica de filtragem muito agressiva é utilizada;

2. Aguçamento (*enhancement*): consiste em aumentar a intensidade dos pixeis que possuem uma mudança significativa com o objetivo de facilitar a detecção das mesmas;

3. Detecção: processo onde as bordas são efetivamente detectadas. O problema é que muitas vezes existem pixeis com valores de gradiente diferentes de zero. Sendo assim, estes nem sempre são considerados como bordas para uma determinada aplicação. Por isso, a escolha do método correto é importante.

2.11 EXTRAÇÃO DE CARACTERÍSTICAS

2.11.1 *Gray-Level Co-Occurence Matrix*

Um dos grandes problemas da área de análise de imagem é como avaliar a diferença de textura. Estas diferenças geralmente variam de acordo com a posição dos pixeis de diferentes intensidades. Uma maneira de descrever as diferenças no relacionamento espacial dos pixeis é utilizando a *Gray-Level Co-Occurrence Matrix*, também conhecida como GLCM (HONEYCUTT; PLOTNICK, 2008).

Inicialmente chamada de *Gray-Tone Spatial-Dependence Matrices*, foi proposta por Haralick, Shanmugam e Dinstein (1973). Os autores afirmam que a textura é uma das mais importantes características utilizadas pra identificar objetos ou regiões de interesse em uma imagem. Por isso, foi proposto um método geral para extrair propriedades texturais.

2.11.1.1 Construção da GLCM

Supondo que $P_\delta(i, j)$ seja uma matriz de co-ocorrência (do inglês, *co-occurence matrix* ou *CM*) e que G seja uma matriz de referência. A matriz de co-ocorrência descreve a frequência com que um elemento g com valor i tem um vizinho com o valor j, em determinada medida de distância angular, δ. Quando se utiliza a GLCM no contexto de processamento de imagens, a matriz de referência I é uma imagem e em cada posição (i, j) existe um *pixel* com determinado valor de nível de cinza (HONEYCUTT; PLOTNICK, 2008).

Cada posição (i, j) da GLCM representa o somatório do número de vezes que um *pixel* com valor i se encontra a uma distância δ de outro com intensidade j (HONEYCUTT; PLOTNICK, 2008). A quantidade de linhas e colunas é determinada pelo número de níveis de cinza presentes na imagem. Por exemplo, se a imagem possui valores de níveis de cinza variando entre 0 e 255, existem 256 níveis. Assim, a GLCM formada terá as dimensões 256×256. Se não houver textura em uma imagem, a matriz resultante vai possuir valores diferentes de zero apenas na diagonal. Na medida em que a quantidade de textura aumenta, os valores fora da diagonal se tornam maiores (BHARATI; LIU; MACGREGOR, 2004).

Supondo uma imagem I, com dimensões $N \times N$, a matriz de

co-ocorrência $P_\delta(i,j)$ pode ser definida como:

$$P_\delta(i,j) = \sum_{x=1}^{N}\sum_{y=1}^{N} \begin{cases} 1, & \text{se } I(x,y) = i \text{ e } I(x+\Delta_x, y+\Delta_y) = j \\ 0, & \text{caso contrário.} \end{cases} \tag{2.6}$$

Onde x e y são os índices das posições dos pixeis, Δ_x e Δ_y descrevem a distância entre o *pixel* que está sendo analisado e o seu vizinho (ELEYAN; DEMIREL, 2011).

Para entender como a GLCM é construída, será utilizada como exemplo uma imagem binária (os pixeis possuem apenas os valores 0 e 1). Ela pode ser descrita por uma matriz I como a que pode ser vista abaixo:

$$I = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Neste exemplo, será considerada a distância entre os pixeis como sendo 1 e o ângulo igual a 90°. Logo, um *pixel* na posição (x,y) terá seu valor comparado com um segundo localizado na posição $(x+1, y+1)$. Ou seja, o vizinho imediato a sua direita.

$$P_{1,90} = \begin{bmatrix} 6 & 0 \\ 0 & 6 \end{bmatrix}$$

Levando em conta a posição $(0,0)$ na matriz acima, tem-se o valor 6. Pode-se interpretar isto como sendo a quantidade de vezes em que o *pixel* analisado (neste caso, ele tem o valor de nível de cinza igual a zero) possui um vizinho logo a direita (distância 1 e ângulo de 90°) com valor zero. Ou seja, o *pixel* em análise tem o valor zero e seu vizinho imediato a direita possui o mesmo valor. Na posição $(1,1)$ da matriz, o valor 6 é novamente encontrado. Isso acontece pelo mesmo motivo.

Analisando as posições $(0,1)$ e $(1,0)$, ambas possuem o valor zero. Isso acontece porque não existem, na imagem I, pixeis com intensidade zero seguidos por pixeis com valor de intensidade igual a 1 e vice-versa. Além disso, na imagem I, existem 2 valores de nível de cinza. Por isso, a GLCM resultante possui dimensões 2×2 (HONEYCUTT; PLOTNICK, 2008).

2.11.1.2 Algoritmo utilizado para Extração de *Features*

Utilizando a GLCM extraída das imagens, podem ser calculadas várias características para descrever a textura nas mesmas. Haralick, Shanmugam e Dinstein (1973) introduziram 14 *features* estatísticas. Estas, são calculadas à partir de quatro GLCMs obtidas nas direções 0°, 45°, 90° e 135° e fazendo uma média entre os quatro valores obtidos. O valor da distância (Δ) entre *pixels* pode ser escolhido. Porém, geralmente utiliza-se "1"(ELEYAN; DEMIREL, 2011).

O algoritmo de referência utilizado para a implementação em OpenCV foi codificado por Uppuluri (2008) utilizando o MATLAB. Neste, além das 14 *features* propostas por Haralick, Shanmugam e Dinstein (1973), o autor extrai mais 9, totalizando 23 características calculadas a partir da GLCM. As *features* excedentes foram propostas por Soh e Tsatsoulis (1999) e Clausi (2002).

Nas fórmulas abaixo, i e j são índices na matriz, N_g é a quantidade de níveis de cinza que a imagem contém e $p(i,j)$ é o valor de nível de cinza do *pixel* na posição (i,j). As características computadas por esse algoritmo são:

1. *Autocorrelation*:
$$f_1 = \sum_i \sum_j (ij)p(i,j) \tag{2.7}$$

2. *Contrast*:
$$f_2 = \sum_{n=0}^{N_g-1} n^2 \left\{ \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j) \middle| |i-j| = n \right\} \tag{2.8}$$

3. *Correlation* (MATLAB)[1]:
$$f_3 = \sum_i \sum_j \frac{(i-\mu_i)(j-\mu_j)p(i,j)}{\sigma_i \sigma_j} \tag{2.9}$$

4. *Correlation*:
$$f_4 = \frac{\sum_i \sum_j (ij)p(i,j) - \mu_i \mu_j}{\sigma_i \sigma_j} \tag{2.10}$$

[1]Quando for vista a notação "MATLAB", significa que tal característica possui duas formas de ser calculada: a proposta pelo autor e a utilizada no *software*. Neste trabalho, ambas serão calculadas.

5. *Cluster Prominence*:

$$f_5 = \sum_i \sum_j (i + j - \mu_i - \mu_j)^4 p(i,j) \tag{2.11}$$

6. *Cluster Shade*:

$$f_6 = \sum_i \sum_j (i + j - \mu_i - \mu_j)^3 p(i,j) \tag{2.12}$$

7. *Dissimilarity*:

$$f_7 = \sum_i \sum_j |i - j| \cdot p(i,j) \tag{2.13}$$

8. *Energy*:

$$f_8 = \sum_i \sum_j p(i,j)^2 \tag{2.14}$$

9. *Entropy*:

$$f_9 = -\sum_i \sum_j p(i,j) \cdot log(p(i,j)) \tag{2.15}$$

10. *Homogeneity* (MATLAB):

$$f_{10} = \sum_i \sum_j \frac{p(i,j)}{1 + |i - j|} \tag{2.16}$$

11. *Homogeneity*:

$$f_{11} = \sum_i \sum_j \frac{p(i,j)}{1 + (i - j)^2} \tag{2.17}$$

12. *Maximum Probability*:

$$f_{12} = MAX_{i,j}\ p(i,j) \tag{2.18}$$

13. *Sum of Squares: Variance*

$$f_{13} = \sum_i \sum_j (i - \mu)^2 p(i,j) \tag{2.19}$$

14. *Sum Average*:

$$f_{14} = \sum_{i=2}^{2N_g} i \cdot p_{x+y}(i) \tag{2.20}$$

15. *Sum Variance*:

$$f_{15} = \sum_{i=2}^{2N_g} (i - f_{16})^2 p_{x+y}(i) \tag{2.21}$$

16. *Sum Entropy*[2]:

$$f_{16} = -\sum_{i=2}^{2N_g} p_{x+y}(i) \cdot log(p_{x+y}(i)) \tag{2.22}$$

17. *Difference Variance*:

$$f_{17} = \sum_{i=1}^{N_g} i^2 \cdot p_{x-y}(i) \tag{2.23}$$

18. *Difference Entropy*:

$$f_{18} = -\sum_{i=0}^{N_g-1} p_{x-y}(i) \cdot log(p_{x-y}(i)) \tag{2.24}$$

19. *Information Measures of Correlation* (1):

$$f_{19} = \frac{HXY - HXY1}{max\{HX, HY\}} \tag{2.25}$$

20. *Information Measures of Correlation* (2):

$$f_{20} = \sqrt{1 - exp\,[-2.0(HXY2 - HXY)]} \tag{2.26}$$

[2]Como existe a probabilidade de alguns resultados darem zero e $log(0)$ não é definido, é recomendado utilizar um termo infinitesimal ϵ ($log(p+\epsilon)$) para os cálculos.

21. *Inverse Difference*:

$$f_{21} = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} \frac{p(i,j)}{1+|i-j|} \tag{2.27}$$

22. *Inverse Difference Normalized*:

$$f_{22} = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} \frac{p(i,j)}{1+|i-j|^2/N_g} \tag{2.28}$$

23. *Inverse Difference Moment Normalized*:

$$f_{23} = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} \frac{p(i,j)}{1+(i-j)^2/N_g} \tag{2.29}$$

Define-se HX e HY como a entropia de p_x e p_y, respectivamente.

$$p_x(i) = \sum_{j=1}^{N_g} P(i,j) \tag{2.30}$$

$$p_y(j) = \sum_{i=1}^{N_g} P(i,j) \tag{2.31}$$

$$\mu = \frac{\sum_i \sum_j P(i,j)}{N_g^2} \tag{2.32}$$

$$\mu_i = \sum_i \sum_j i \cdot p(i,j) \tag{2.33}$$

$$\mu_j = \sum_i \sum_j j \cdot p(i,j) \tag{2.34}$$

$$\sigma_i = \sum_i \sum_j (i-\mu_i)^2 \cdot p(i,j) \tag{2.35}$$

$$\sigma_j = \sum_i \sum_j (i-\mu_j)^2 \cdot p(i,j) \tag{2.36}$$

$$HXY = -\sum_i \sum_j p(i,j) \cdot log(p(i,j)) \tag{2.37}$$

$$HXY1 = -\sum_{i}\sum_{j} p(i,j) \cdot log(p_x(i)p_y(j)) \tag{2.38}$$

$$HXY2 = -\sum_{i}\sum_{j} p_x(i)p_y(j) \cdot log(p_x(i)p_y(j)) \tag{2.39}$$

$$p_{x+y}(k) = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} p(i,j),\ i+j=k\ e\ k=2,3,...,2N_g. \tag{2.40}$$

$$p_{x-y}(k) = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} p(i,j),\ |i-j|=k\ e\ k=0,1,...,N_g-1. \tag{2.41}$$

2.11.2 *Segmentation-based Fractal Texture Analysis* (SFTA)

A textura nos dá informações sobre distribuição espacial de cor ou intensidades em uma imagem completa ou uma região desta. Porém, a extração das informações de textura consomem muito tempo (ARIVAZHAGAN et al., 2015). Isto acontece geralmente em casos onde a utilização de imagens de alta resolução é indispensável para manter a alta taxa de acerto. Métodos como o GLCM podem ser utilizados. Porém, precisam de um grande poder computacional (SAMIAPPAN et al., 2017). É neste contexto que entra o *Segmentation-based Fractal Texture Analysis* (SFTA) (COSTA; HUMPIRE-MAMANI; TRAINA, 2012).

Este algoritmo pode ser dividido em duas partes. Na primeira, a imagem em escala de cinza é decomposta em um conjunto de imagens binárias utilizando o *Two-Threshold Binary Decomposition* (TTBD), proposto pelo mesmo autor. Para cada imagem resultante, as *fractal dimensions* dos limites de cada região são calculadas. Além disso, a média do nível de cinza e uma contagem de pixeis são feitas.

O TTBD utiliza uma imagem de entrada $I(x, y)$ e retorna um conjunto de imagens binárias. O primeiro passo é calcular um intervalo T com vários valores de *thresholds*. Estes são obtidos pela seleção de níveis de cinza espaçados. Isto pode ser conseguido pela utilização do *Multi-Level Otsu Algorithm* (LIAO et al., 2001) aplicado recursivamente n_t vezes. A quantidade de vezes (n_t) é um parâmetro definido pelo usuário e influencia no tamanho do vetor de características. Uma imagem binária $I_b(x, y)$ é obtida aplicando-se a Equação 2.42.

$$I_b(x,y) = \begin{cases} 1, & \text{se } t_l < I(x,y) \leq t_u \\ 0, & \text{outro caso.} \end{cases} \tag{2.42}$$

t_l representa limite de nível de cinza inferior e t_u representa o limite superior.

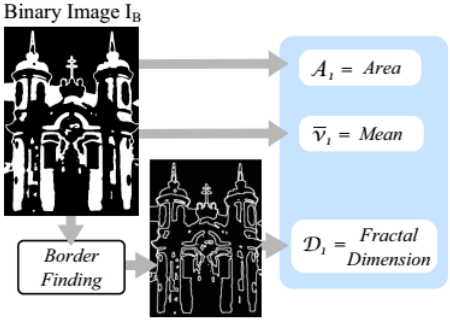
O conjunto de imagens binárias é obtido aplicando-se a Equação 2.42 na imagem de entrada, utilizando todos os pares contínuos de *thresholds* de $T \cup \{n_l\}$ e todos os pares de *thresholds* $\{t, n_l\}, t \in T$, onde n_l corresponde ao máximo valor possível de nível de cinza em $I(x,y)$. Este processo retorna $2n_t$ imagens.

Na segunda parte do algoritmo, um vetor contendo a quantidade de pixeis, a média do nível de cinza e as *fractal dimensions* das imagens binárias é criado. As duas primeiras características são calculadas a partir da imagens binárias geradas pelo TTBD. As *fractal dimensions* são obtidas através de uma imagem binária $I_b(x,y)$ e é representada por uma imagem $\Delta(x,y)$ que contém só bordas. Esta, representa os limites das regiões de cada imagem e é calculada por:

$$\Delta(x,y) = \begin{cases} 1, \text{se } \exists (x',y') \in N_8[(x,y)] : \\ \quad I_b(x',y') = 0 \wedge I_b(x,y) = 1, \\ 0, \text{outro caso.} \end{cases} \tag{2.43}$$

$N_8[(x,y)]$ representa um conjunto de 8 pixeis que estão conectados a (x,y). $\Delta(x,y)$ é "1" se $I_b(x,y) = 1$ e, pelo menos, um se seus vizinhos for igual a zero. A imagem resultante possui bordas de um *pixel* de largura.

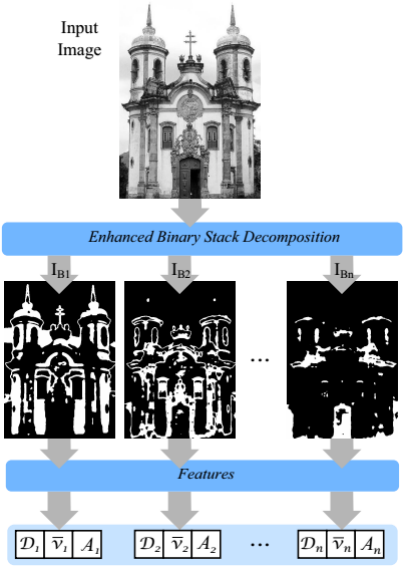
Figura 9 – Resumo do algoritmo aplicado



Extraído de: (COSTA; HUMPIRE-MAMANI; TRAINA, 2012)

A média de nível de cinza e a contagem de pixeis traz a vantagem de complementar a informação extraída sem aumentar muito o tempo computacional. O tamanho do vetor de características é três vezes a quantidade de imagens binárias produzidas pelo TTBD. Isto acontece porque de cada imagem binária são extraídas três características: a quantidade de pixeis, a média de nível de cinza e as *fractal dimensions*.

Figura 10 – Características extraídas de cada imagem



Extraído de: (COSTA; HUMPIRE-MAMANI; TRAINA, 2012)

As *fractal dimensions* podem ser eficientemente calculadas em uma quantidade linear de tempo utilizando o *Box Counting Algorithm* (TRAINA et al., 2000). Logo, a complexidade assintótica do SFTA é $O(N \cdot |T|)$, onde N representa a quantidade de pixeis da imagem em escala de cinza e $|T|$ é o número de diferentes *thresholds* do *Multi-Level Otsu Algorithm*.

3 APRENDIZADO DE MÁQUINA

Desde a época em que o computador foi inventado, vêm-se imaginando o que ele pode aprender. Se for descoberta uma maneira de fazer com que ele realize esta tarefa como os seres humanos, através da experiência, o impacto seria enorme. No momento em que isto acontecer, o uso do computador subiria a um novo nível (MITCHELL, 1997).

Até o momento, algoritmos foram criados e certas tarefas de aprendizado são bem executadas. É muito difícil escrever algoritmos para realizar alguns trabalhos que são facilmente feitos pelos seres humanos, como reconhecer pessoas pelo seu rosto (FACELI et al., 2011). No momento que o conhecimento sobre esta área se tornar maduro, o aprendizado de máquina terá um dos papéis principais na ciência da computação (MITCHELL, 1997).

Na década de 1970, a área de inteligência artificial começou a ser utilizada para a resolução de problemas reais. Com o aumento da complexidade destes e a crescente geração de dados, tornou-se necessário criar soluções mais sofisticadas, que fossem capazes de aprender por experiências passadas. Estas técnicas deveriam conseguir criar uma hipótese ou função capaz de resolver o problema que se deseja tratar. A este processo dá-se o nome de Aprendizado de Máquina (FACELI et al., 2011).

Faceli et al. (2011) afirmam que essa capacidade de aprendizado é essencial para que haja um comportamento inteligente. Segundo Mitchell (1997), uma boa parte do aprendizado consiste em adquirir informações de exemplos de treinamento. O mesmo autor ainda define AM como sendo "a capacidade de melhorar o desempenho na realização de alguma tarefa por meio da experiência".

Apesar de ser naturalmente associada a área de inteligência artificial, muitas outras têm contribuição direta para o avanço da AM. Entre elas, podemos citar Probabilidade e Estatística, Teoria da Informação e até Neurociência. O aprendizado de máquina é uma das áreas que mais vem crescendo nos últimos anos (FACELI et al., 2011).

3.1 CLASSIFICAÇÃO

No contexto de processamento de imagens, o objetivo da classificação é identificar características, padrões ou estruturas em uma imagem e utilizá-las para atribuir um objeto ou a própria imagem à

uma determinada classe (SOLOMON; BRECKON, 2011).

A classificação autônoma é um campo que pertence mais ao reconhecimento de padrões do que ao processamento de imagens. Porém, esta etapa é essencial para projetos desta área (SOLOMON; BRECKON, 2011).

A grande dificuldade de criar um sistema de classificação é que, muitas vezes, o relacionamento entre os parâmetros e os dados contidos na imagem não são claros. Os objetos a serem classificados não estão diretamente relacionados a uma faixa de valores referentes a uma única característica. Então é necessário que eles sejam identificados de acordo com as características que mais o distinguem do restante da imagem (JÄHNE, 2002).

Isso requer uma investigação cuidadosa. Deve-se escolher tais características onde as classes dos objetos possam ser distinguidas com o mínimo de esforço pelo sistema, associado a técnica correta de classificação (JÄHNE, 2002).

Apesar de o ser humano conseguir classificar a maioria dos objetos de modo mais preciso, existem tarefas onde utilizar um sistema de classificação automatizado é necessário. Por exemplo, quando existe um volume de dados muito grande. Um sistema computacional consegue fazer esta classificação em um tempo muito menor do que os seres humanos (SOLOMON; BRECKON, 2011).

Solomon e Breckon (2011) ainda afirmam que durante a etapa de classificação existem duas tarefas em que o desenvolvedor ainda precisa interferir. São elas:

- Especificação da tarefa: o desenvolvedor do sistema precisa definir exatamente quais classes serão consideradas e quais varáveis ou parâmetros serão importantes para alcançar este objetivo;

- *Class labeling*: no aprendizado supervisionado, o desenvolvedor precisa indicar (manualmente) a qual classe um determinado exemplo pertence, baseado em suas propriedades.

3.1.1 Aprendizado Supervisionado e Não-Supervisionado

As técnicas de classificação podem ser divididas em duas classes principais: supervisionadas e não-supervisionadas. A primeira, precisa de um conjunto de treinamento onde o vetor de características (*features vector*) possua a indicação de qual classe este pertence. Utilizando estes dados como conjunto de treinamento, tem-se como objetivo fa-

zer com que os novos exemplos possam ser corretamente classificados baseado no que foi aprendido na fase anterior. Espera-se criar um sistema de classificação que consiga *generalizar bem* para novos exemplos (SOLOMON; BRECKON, 2011).

Já no aprendizado não-supervisionado não existem exemplos que indicam a qual classe um determinado vetor de características pertence. Espera-se que os grupos possam ser identificados baseando-se no conjunto de treinamento como um todo e as características que permitam distinguir um grupo de outro (SOLOMON; BRECKON, 2011).

3.2 REDES NEURAIS ARTIFICIAIS

Uma parte da motivação para o estudo das Redes Neurais Artificiais (RNAs) veio da observação do sistema de aprendizado biológico. Estes, são construídos através de redes complexas de neurônios (MITCHELL, 1997).

Uma rede neural é uma maneira distribuída e paralela de processamento de informações. Ela é formada pelos elementos de processamento, ligados por um canal de sinal chamado de "conexão". Cada elemento de processamento tem uma conexão de saída única, onde esta saída pode ser espalhada para quantas direções forem desejadas. O sinal de saída pode ser de qualquer tipo matemático (HECHT-NIELSEN et al., 1988).

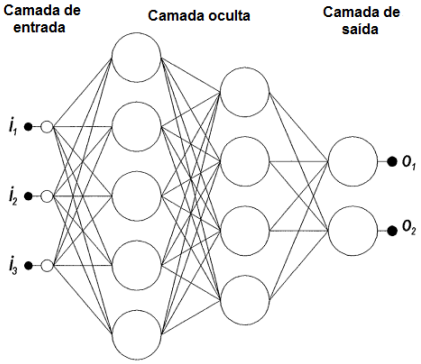
3.2.1 *Multilayer Perceptron*

Uma rede neural do tipo *Multilayer Perceptron* (MLP) consiste em um sistema de neurônios interconectados. Este, representa o mapeamento não-linear entre a entrada e a saída. Os neurônios estão conectados por ligações ponderadas. Ou seja, cada ligação possui um valor multiplicador que dará peso a conexão. A saída é uma função da soma das entradas modificadas por uma função de ativação (ou função de transferência) (GARDNER; DORLING, 1998).

O saída dos nós são ponderadas pelo peso da conexão e estes valores de saída são a entrada para a próxima camada da rede. Isso demonstra a direção de processamento da informação, já que as MLPs são conhecidas como redes neurais *feed forward*. Ou seja, a informação é encaminhada sempre em uma direção (GARDNER; DORLING, 1998). A estrutura básica de uma Rede Neural *Multilayer Perceptron* é mostrada

na Figura 11.

Figura 11 – Rede Neural Artificial - *Multilayer Perceptron*



Fonte: Adaptada de Gardner e Dorling (1998).

As MLPs podem ter uma ou mais camadas ocultas, antes da camada de saída. Uma rede neural pode ser "completamente conectada". Isso acontece quando cada neurônio é conectado com todos os outros da camada anterior e da próxima (GARDNER; DORLING, 1998).

As redes neurais são métodos de aprendizado supervisionado. Assim, ela "aprende" através de treinamento e isso requer um conjunto de dados. Durante esta fase, a rede é alimentada repetidamente com os dados de treinamento e os pesos das conexões são ajustados até que o conjunto de entradas e saídas estejam de acordo com o esperado (GARDNER; DORLING, 1998).

Durante a fase de treinamento, as saídas podem não ser iguais as desejadas. O erro é definido pela diferença entre a saída atual e a desejada. Durante esta fase, a magnitude do erro é utilizada para determinar quanto os pesos das conexões devem ser ajustados para que este valor de erro possa ser diminuído. Sendo assim, existem vários algoritmos que podem ser utilizados para treinamento de uma rede neural (GARDNER; DORLING, 1998). Neste trabalho, utilizou-se o *Backpropagation*.

3.2.1.1 *Backpropagation*

Backpropagation é um método utilizado para o treinamento de redes neurais. Tem como objetivo calcular a contribuição do erro de

cada neurônio depois que uma quantidade de dados é processada. É utilizado para ajustar os pesos de cada neurônio, completando o processo de aprendizado para cada caso (NIELSEN, 2015). Este algoritmo é muito popular por ser conceitualmente simples e computacionalmente eficiente (LECUN et al., 2012).

Tecnicamente falando, é necessário calcular o gradiente da função de perda. O algoritmo requer que a saída desejada para cada entrada seja conhecida. Por esse motivo, pode ser considerado como um algoritmo de aprendizado supervisionado (NIELSEN, 2015).

Nielsen (2015) ainda afirma que o algoritmo de *backpropagation* vem sendo redescoberto. Está relacionado ao algoritmo de *Gauss-Newton* e é parte da pesquisa contínua na área de redes neurais.

3.3 *SUPPORT VECTOR MACHINE*

Os *Support Vector Machines* (SVMs) são um dos algoritmos de aprendizado de máquina mais populares. Estes são utilizados para classificação, regressão e outras tarefas de aprendizado (CHANG; LIN, 2011). Apesar de este método ser considerado mais fácil de usar do que as redes neurais, pessoas que não são familiarizadas com ele acabam tendo resultados ruins de início. O objetivo do SVM é produzir um modelo, baseado nos dados de treinamento, que é capaz de prever a classe da amostra utilizando apenas seus atributos (HSU et al., 2003).

Utilizando os dados de treinamento, organizados como "amostra-classe" $(x_i, y_i),\ i = 1, ..., l$, onde $x_i \in R^n$ e $y \in \{1, -1\}^l$, o SVM procura por uma solução para o problema de otimização descrito por:

$$\begin{aligned} \min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \\ \text{Sujeito à } y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i, \\ \xi_i \geq 0. \end{aligned} \tag{3.1}$$

Os dados de treinamento x_i são mapeados para um espaço dimensional maior (podendo ser infinito) pela função ϕ. O SVM tenta encontrar um hiperplano capaz de separar os dados linearmente, com margem máxima, neste novo espaço dimensional (HSU et al., 2003).

O *Support Vector Machine* é um algoritmo de classificação baseado em *kernel*. Basicamente, existem quatro tipos: o linear, o polinomial, o sigmoide e o *radial basis function*. Para este trabalho, o último

foi escolhido por lidar melhor com dados não-lineares (HSU et al., 2003). A Equação 3.2 descreve o *kernel* utilizado.

$$K(x_i, x_j) = exp(-\gamma||x_i - x_j||^2), \gamma > 0 \tag{3.2}$$

Existem dois parâmetros principais: C e γ. Estes devem ser escolhidos de modo apropriado. O parâmetro C representa o custo da penalidade. A escolha deste parâmetro influencia nos resultados da classificação. Se for muito grande, a taxa de acerto pode ser muito alta durante a fase de treinamento e muito baixa no momento da predição. Já γ, se for muito alto, pode gerar *overfitting*. Se for muito baixo, pode levar a *underfitting* (LIN et al., 2008).

3.3.1 *Grid Search Algorithm*

Como citado anteriormente, existem dois parâmetros principais para o SVM com *kernel* do tipo RBF, C e γ. Inicialmente, seus valores ótimos para determinada aplicação não são conhecidos. Sendo assim, algum tipo de busca deve ser feito. O objetivo é encontrar o par de valores que traz uma boa taxa de acerto no momento da predição (HSU et al., 2003).

Para este trabalho, utilizou-se o *Grid Search Algorithm* (HSU et al., 2003). Juntamente a este, foi utilizada validação cruzada (*cross-validation*), visando evitar o problema de *overfitting*. O algoritmo consiste em tentar vários pares (C, γ) e escolher aquele que traz a melhor acurácia utilizando *cross-validation*. Isto é feito testando sequências exponencialmente crescentes.

Hsu et al. (2003) ainda afirma que este método é simples de entender e pode parecer "ingênuo". Porém, seu desempenho se equipara com algoritmos mais complexos, já que neste caso procura-se por apenas dois parâmetros.

O *Grid Search* pode consumir muito tempo computacional. Por isso, recomenda-se utilizar um *coarse-grid* primeiro. Assim, depois que for identificada a melhor região do *grid*, uma busca mais específica pode ser feita (HSU et al., 2003).

Inicialmente, utiliza-se os dados de treinamento para encontrar o par de parâmetros que traz o melhor resultado. Quando isso ocorre, utiliza-se novamente estes dados para treinar o classificador. Então, com os dados de teste, chega-se ao resultado através do classificador previamente treinado.

3.4 ÁRVORES DE DECISÃO

Uma Árvore de Decisão pode ser entendida como um procedimento de classificação que particiona recursivamente o conjunto de dados. Este é dividido em subdivisões menores com base no conjunto de testes definido em cada nó da árvore. Esta, é composta por um nó raiz, que representa todos os dados. Além disso, possui uma série de nós internos, gerados pela divisão de seus nós pais. Por último, existem os nós terminais (ou folhas), caracterizados por não possuírem nós filhos. Cada nó em uma árvore de decisão possui apenas um nó pai (FRIEDL; BRODLEY, 1997).

A árvore de decisão se comporta como uma árvore binária. A segunda é caracterizada pelo fato de cada nó possuir dois nós filhos. As árvores de decisão podem ser utilizadas para classificação ou regressão. Para a primeira atividade, cada folha é caracterizada como uma classe e muitas folhas podem ter a mesma classe (BREIMAN et al., 1984).

3.4.1 Treinamento

A árvore de decisão é construída recursivamente, começando pelo nó raiz. Todos os dados de treinamento são utilizados para separá-lo. Em cada nó, a regra de decisão ótima é encontrada baseada em algum critério. Podem ser utilizados critérios diferentes quando se trata de classificação ou regressão (BREIMAN et al., 1984).

Os dados são divididos em conjuntos a cada nó. Este processo é repetido recursivamente, até que algum critério de parada seja atingido. São exemplos deste:

- A profundidade da árvore atingiu o seu valor máximo definido;
- A quantidade de amostras de treinamento nos nós é menor do que um valor específico, fazendo com que não seja mais necessário a divisão destes dados;
- Todos os dados no nó pertencem a mesma classe (ou quando a variação entre eles é muito pequena, em caso de regressão).

Quando a árvore é construída, esta deve ser "podada" utilizando validação cruzada, se necessário. Isso previne que o modelo seja levado a um *overfitting* (BREIMAN et al., 1984).

3.4.2 Predição

Para chegar ao nó folha e obter a resposta relacionada ao vetor de característica de entrada, o procedimento de predição deve começar no nó raiz. Para cada nó "não folha", o algoritmo se move para esquerda ou para direita, tendo como base o valor de certa variável (BREIMAN et al., 1984).

Existem dois tipos de variáveis mais comuns. As primeiras, são as Variáveis Ordenadas. Elas são comparadas com o valor do nó. Se este for menor, o algoritmo se move para a esquerda. Se for maior, para a direita.

O segundo tipo é conhecida como Variável Categórica. Nestas, valores discretos são testados para verificar se estes pertencem a um certo subconjunto de valores. Se pertencer, o algoritmo se move para a esquerda. Senão, para a direita (BREIMAN et al., 1984).

3.5 *RANDOM TREES*

As *Random Trees* (também conhecidas como *Random Forests*) são uma coleção de árvores de decisão. Esta coleção é chamada de *forest* (floresta). A classificação funciona através de uma espécie de "votação". Cada árvore na floresta tem como entrada um vetor de características e estes são classificados individualmente. A resposta final é a aquela que receber a maioria dos "votos". No caso de regressão, a resposta final é a média das respostas de todas as árvores na floresta (BREIMAN; CUTLER, 2003).

Todas as árvores são treinadas com os mesmos parâmetros, utilizando diferentes subconjuntos de treinamento. Cada subconjunto é gerado a partir do conjunto de treino original. Nem todas as variáveis são utilizadas para encontrar a melhor separação nos nós das árvores treinadas. É utilizado apenas um subconjunto aleatório destas variáveis.

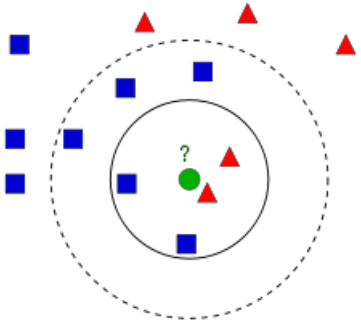
A cada nó, um novo subconjunto é gerado. Porém, o seu tamanho é fixo em todas as árvores. Nenhuma destas é podada. Quando se utiliza *Random Trees*, não há necessidade de utilizar um método para estimar a acurácia, já que o erro é estimado internamente durante o treinamento (BREIMAN; CUTLER, 2003).

3.6 *K-NEAREST NEIGHBOR*

O *k-Nearest Neighbor* (Algoritmo do Vizinho Mais Próximo, em português) é o algoritmo mais básico que funciona a base de exemplos. Este assume que todos os dados correspondem a pontos em um espaço n-dimensional $\mathbb{R}^n$. Os vizinhos de uma determinada amostra são definidos em termos da Distância Média Euclidiana (MITCHELL, 1997).

De modo prático, precisa-se entender que cada amostra está presente em um espaço de características (do inglês, *feature space*) e pertence a uma classe. O modo mais simples de classificação é verificar qual é o vizinho mais próximo desta amostra (UMANAND, 2012).

Figura 12 – Exemplo de *feature space* do kNN



Fonte: Adaptada de OpenCV (2014).

Para entender melhor, deve-se levar em conta a situação mostrada na Figura 12. Uma amostra desconhecida (círculo verde) é inserida no espaço e sua classe é desconhecida. Se for analisado apenas o vizinho mais próximo, ela é classificada como sendo um triângulo vermelho. Se forem analisado três vizinhos, ela continuará sendo inserida na mesma classe.

Porém, se forem analisados cinco vizinhos, esta amostra de classe desconhecida será classificada como sendo um quadrado azul. Este fato mostra que a classificação de determinado item depende do valor k de vizinhos analisados. Se este valor for par, pode haver empate. Por isso, recomenda-se escolher k como um valor ímpar (UMANAND, 2012).

Supondo $k = 4$, pode acontecer de, próxima da amostra de classe desconhecida, haverem dois triângulos vermelhos. Um pouco mais afastadas, estão dois círculos azuis. Teoricamente, isso seria um empate.

Uma maneira de decidir a qual classe a amostra desconhecida pertence é dar pesos a cada família, de acordo com a sua distância da amostra em análise. Aquelas que estão mais próximas, recebem um peso maior. A amostra em análise será colocada na classe que possuir o maior valor total de pesos. Este método é conhecido como kNN modificado (UMANAND, 2012).

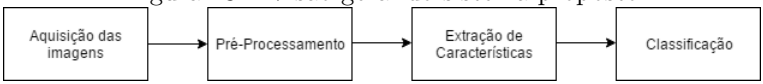
É importante notar que, quando há um número muito grande de amostras e classes, é necessário muita memória para classificar as amostras em análise. Porém, é importante ressaltar que o tempo de treinamento necessário é quase nulo (UMANAND, 2012).

4 SISTEMA DE CLASSIFICAÇÃO DE PISOS CERÂMICOS

Na literatura, muitas abordagens foram propostas tendo como objetivo resolver os problemas presentes em uma linha de produção na indústria cerâmica. Neste trabalho, utilizando técnicas das áreas de processamento de imagens e aprendizado de máquina, propõe-se uma abordagem completa para verificação de defeitos visuais em pisos cerâmicos.

Segundo Pedrini e Schwartz (2008), um sistema de processamento digital de imagens é constituído por um conjunto de etapas capazes de produzir um resultado a partir do *domínio do problema*. Sendo assim, escolheu-se utilizar um sistema com as etapas semelhantes às propostas pelo autor. Estas, podem ser vistas na Figura 13.

Figura 13 – Visão geral do sistema proposto



Fonte: Elaborada pelo autor

A primeira etapa consiste na aquisição das imagens contendo o piso a ser analisado. Esta precisa ser capturada em um ambiente controlado, com o mínimo possível de influência externa. Para isto, foi utilizado um protótipo de baixo custo, que será explicado durante o capítulo.

Na etapa de pré-processamento, a informação que não é necessária no momento da análise do piso é removida. Por exemplo, a imagem capturada mostra uma parte do fundo do protótipo, juntamente com o sistema de iluminação. Como estas informações não serão analisadas, podem ser removidas. Fazendo isto, a quantidade de informação a ser processada é diminuída.

Na etapa de extração de características são utilizados os algoritmos de análise de textura. Esta fase tem como principal objetivo adquirir informações estatísticas que possam descrever o piso em análise.

A última etapa consiste na classificação da placa cerâmica. Neste trabalho serão comparados vários classificadores, com diferentes características. Tem-se o intuito de verificar qual é a melhor combinação entre algoritmo de extração e classificador para este tipo de aplicação.

Os algoritmos foram desenvolvidos na linguagem C++, utili-

zando a biblioteca OpenCV. Nenhuma otimização foi feita.

4.1 OPENCV

O OpenCV (*Open Source Computer Vision*) é um conjunto de bibliotecas que possui várias funções voltadas à aplicações que envolvem visão computacional. Estas, tiram vantagem da utilização de múltiplos *cores*. Além disso, a biblioteca é muito utilizada em aplicações de tempo real (BRADSKI; KAEHLER, 2008).

Um dos principais objetivos do OpenCV é entregar uma infraestrutura que auxilia os desenvolvedores a criar aplicações de visão computacional de modo muito rápido. Existem mais de 500 funções disponíveis, distribuídas em várias áreas (BRADSKI; KAEHLER, 2008).

Segundo Bradski e Kaehler (2008), a área de aprendizado de máquina está intimamente ligada com o processamento de imagens. É disponibilizada também uma biblioteca completa para este fim, chamada de *Machine Learning Library* ou MLL. Esta, tem seu foco principal em reconhecimento de padrões estatísticos e métodos de clusterização.

Escolheu-se utilizar o OpenCV por ser uma biblioteca muito popular e elogiada pelos usuários. O desenvolvimento pode ser feito em várias linguagens de programação, incluindo o C/C++ e Python. Além disso, possui uma vasta documentação, facilitando muito no momento do desenvolvimento. Possui código aberto e pode ser utilizada tanto educacionalmente quanto comercialmente. Ainda é otimizada utilizando bibliotecas de processamento paralelo.

4.2 PROTÓTIPO

Uma aplicação de visão computacional precisa de um sistema de aquisição robusto, onde a captura das imagens possa ser feita de maneira correta, em um ambiente controlado. Neste trabalho, optou-se por criar um protótipo de baixo custo para a aquisição das imagens.

O principal objetivo deste protótipo é proporcionar um ambiente estável e sem interferências externas para adquirir as imagens dos pisos. Na confecção do protótipo, foram levadas em conta as características que seriam importantes em uma linha de produção de uma indústria cerâmica. Entre as principais podemos citar: ambiente estável, boa iluminação e fundo escuro, visando criar um maior contraste com pisos

claros, o padrão mais comum encontrado na indústria.

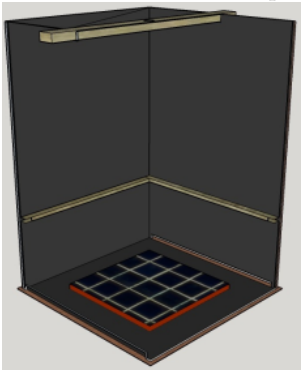
Figura 14 – Visão externa do protótipo

Fonte: Elaborada pelo autor

A visão externa do protótipo é mostrada na Figura 14 e se assemelha a uma grande caixa. Foi feita com *Medium Density Fiberboard* (MDF) e possui as dimensões 50 cm × 50 cm × 70 cm.

Este possui uma entrada e saída para os pisos, além de uma base removível para um melhor posicionamento dos mesmos. Isso também permite a fácil retirada da placa cerâmica depois que a aquisição das imagens é feita. Tem como objetivo manter um ambiente controlado, fazendo com que haja o mínimo de influência externa. A base é removível para permitir a adaptação à uma linha de produção.

Figura 15 – Visão interna do protótipo



Fonte: Elaborada pelo autor

A Figura 15 mostra a parte interna do protótipo. Na parte de cima, há um suporte onde a câmera é fixada. Na parte inferior, existe uma estrutura para colocação do sistema de iluminação. Este, foi feito utilizando uma fita de LED de 3m, contendo 120 LEDs. Foi fixada na parte de baixo do suporte com a intenção de evitar sombras e/ou reflexos provenientes da iluminação.

Seu interior foi pintado de preto fosco com a intenção de criar um contraste maior com os padrões de pisos mais comuns fabricados atualmente: os de tons claros. Uma limitação deste sistema é o fato de só poderem ser analisadas placas cerâmicas com dimensões máximas de 48 cm $\times$ 48 cm. Isto pode ser resolvido se o tamanho do protótipo for aumentado e a câmera for posicionada a uma altura maior.

Esta, possui um sensor T4K37 com 13 *megapixels* CMOS de 1/3.07", posicionada a uma altura de 69 cm em relação ao piso. Com isso, obteve-se um *ground sample distance* (GSD) de aproximadamente 0,0206 cm. Apesar de a câmera não ser a ideal para uma aplicação industrial, ela cumpriu bem o seu objetivo de auxiliar na validação da abordagem proposta. Também deve ser levado em conta o fato de que a ideia era criar um protótipo de baixo custo.

As imagens foram adquiridas com a câmera presente no protótipo de aquisição de imagens. Logo depois, foram transferidas via cabo USB para um computador, onde seguiram o fluxo proposto neste trabalho (pré-processamento, extração de características e classificação).

A acurácia do sistema poderia ser melhorada se houvesse uma câmera apropriada para a aplicação. Porém, isso resultaria em utilizar imagens com uma resolução maior e para isso, seria necessário obter um *hardware* com maior poder de processamento. Assim, a taxa de acerto poderia ser melhorada, mantendo (ou até melhorando) o tempo de processamento.

4.3 BASE DE DADOS

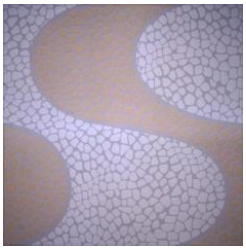
Levando em conta que foram utilizados métodos de aprendizado supervisionado, era necessário criar uma base de dados para o treinamento dos classificadores. Para isso, utilizou-se os recursos disponíveis visando simular a maioria dos defeitos possíveis encontrados em uma linha de produção de uma indústria cerâmica.

A solução foi desenvolvida com a intenção de classificar qualquer tipo de piso. Por isso, optou-se por criar uma base de dados com cinco padrões diferentes, com e sem textura. Estes podem ser vistos na Figura

Figura 16 – Tipos de pisos contidos na base de dados

(a) Tipo 1

(b) Tipo 2



(c) Tipo 3

(d) Tipo 4

(e) Tipo 5

Fonte: Elaborada pelo autor

Tabela 1 – Divisão dos padrões de pisos na base de dados

Tipos	**Quantidades**		
	Bons	Ruins	Total
1	20	141	161
2	47	139	186
3	41	123	164
4	287	116	403
5	300	80	380
Total	695	599	**1294**

Fonte: Elaborada pelo autor

16. As imagens foram obtidas utilizando o protótipo citado na Seção 4.2 e aplicando as operações de pré-processamento.

Os pisos estão em quantidades diferentes e estas podem ser vistas na Tabela 1. Obteve-se um total de 1294 imagens. Os pisos são classificados entre *bons* (sem defeitos) ou *ruins* (com algum tipo de defeito). A classificação de cada placa cerâmica foi feita baseada nas regras apresentadas nas NBRs 13816, 13817 e 13818 (TÉCNICAS, 1997).

É importante salientar que ter pisos com e sem texturas é um ponto positivo. Isso acontece pelo fato de que, desta maneira, é possível analisar como os algoritmos de extração de características se comportam com cada padrão. Para a análise de resultados foram levadas em conta, principalmente, a taxa de acerto e o tempo de processamento necessário.

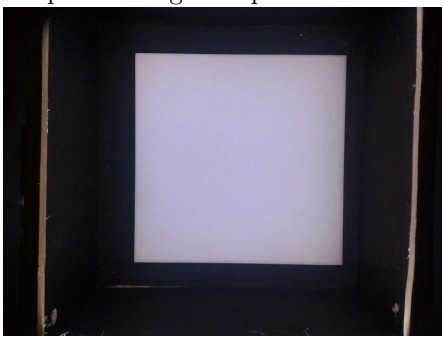
4.4 PRÉ-PROCESSAMENTO

O objetivo da etapa de pré-processamento é remover toda a informação que não será utilizada na etapa de extração de características. Um exemplo de uma imagem adquirida com o protótipo pode ser vista na Figura 17.

Com o intuito de diminuir o tempo de processamento, reduz-se a imagem para 1/5 do seu tamanho original. Dessa maneira, a quantidade de informação a ser processada é reduzida. Isto pode ser feito porque neste tipo de aplicação não é necessária a utilização de imagens de alta resolução para obter bons resultados, se comparados com a literatura.

Depois disto, a matriz que representa a imagem é realocada em um vetor para que possa ser aplicado o *k-means clustering*. A aplicação deste método tem como objetivo segmentar a imagem em *clusters*. Ou

Figura 17 – Exemplo de imagem capturada utilizando o protótipo



Fonte: Elaborada pelo autor

seja, é feita uma segmentação por cor, com o objetivo de "encontrar" o piso na imagem.

Como já foi citado, o protótipo possui o fundo escuro para que haja um maior contraste entre este e o piso cerâmico. Sendo assim, a utilização de dois *clusters* ($k = 2$) para essa aplicação é suficiente.

A Equação 4.1 (OpenCV, 2014) define a função utilizada para divisão dos *clusters*. Para a inicialização de centros, foi utilizado o *k-means++ center initialization* (ARTHUR; VASSILVITSKII, 2007).

$$\sum_{i} ||samples_i - centers_{labels_i}||^2 \tag{4.1}$$

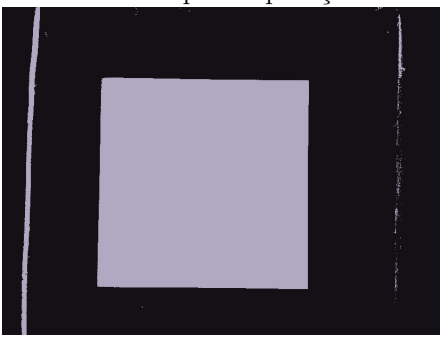
O resultado obtido após a aplicação deste método é mostrado na Figura 18. Pode-se ver claramente que há dois grupos na imagem: um contendo o piso, juntamente com parte do sistema de iluminação e outro contendo o fundo do protótipo.

Uma operação de *thresholding* foi realizada para converter a imagem obtida em binária. O resultado é mostrado na Figura 19.

O próximo passo é encontrar o piso propriamente dito e remover o restante das informações desnecessárias. Para isso, é procurada na imagem a área com a maior quantidade de pixeis. Assim que ela é encontrada, o algoritmo calcula o menor quadrado que pode descrevê-la. O resultado é mostrado na Figura 20.

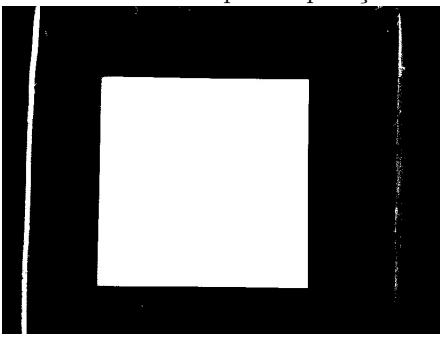
Geralmente, em uma linha de produção, os pisos não chegam dispostos da mesma maneira. Desta forma, pode haver angulação nestes e isso fica claro nas imagens capturadas. Para retirá-las, é necessário calcular o ângulo formado entre os dois vértices inferiores. Quando

Figura 18 – Resultado obtido após a aplicação do *k-means clustering*



Fonte: Elaborada pelo autor

Figura 19 – Resultado obtido após a operação de *thresholding*

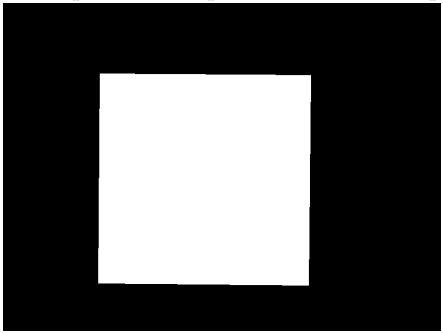


Fonte: Elaborada pelo autor

este valor é encontrado, uma transformada *Affine* é aplicada. Assim, se houver uma angulação, positiva ou negativa, ela será removida. O resultado é mostrado na Figura 21.

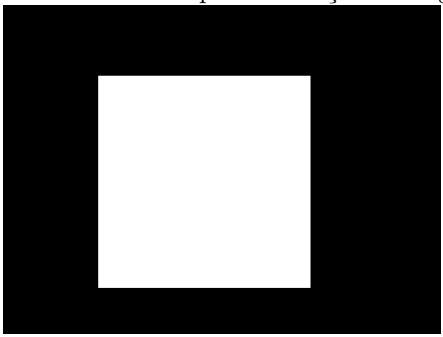
Após o passo anterior, a máscara que descreve o piso está pronta. Aplicando-a na imagem original, obtém-se o piso que será submetido à fase de extração de características. O resultado final da etapa de pré-processamento é mostrado na Figura 22.

Figura 20 – Menor quadrado capaz de descrever o piso em análise



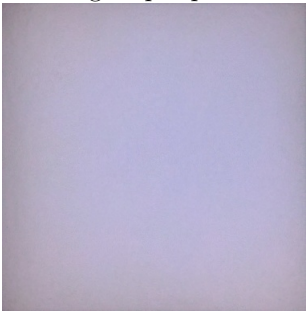
Fonte: Elaborada pelo autor

Figura 21 – Resultado após a remoção da angulação



Fonte: Elaborada pelo autor

Figura 22 – Imagem pré-processada do piso



Fonte: Elaborada pelo autor

4.5 EXTRAÇÃO DE CARACTERÍSTICAS

A textura é uma característica fundamental em uma imagem. A partir dela, pode-se obter informações sobre a distribuição espacial de cores ou intensidades. Desta maneira, estas informações podem ser utilizadas para classificação. Porém, a extração da informação de textura consume muito tempo (ARIVAZHAGAN et al., 2015). Isso acontece principalmente em casos onde a utilização de imagens de alta resolução é indispensável (SAMIAPPAN et al., 2017). Para a extração de características, métodos como o GLCM ou o SFTA podem ser utilizados.

Porém, alguns métodos demandam muito tempo computacional. A escolha do algoritmo de extração é algo que requer muito cuidado, principalmente em casos de aplicações que devem funcionar em tempo real. Este é o caso de uma linha de produção em uma indústria cerâmica. A classificação dos pisos tem de ser feita de modo muito rápido e preciso. Por isso, a escolha do método adequado precisa ser feita.

4.5.1 Utilização da GLCM para extração de características

Um dos métodos escolhidos para extração de características foi o *Gray-Level Co-Occurence Matrix* (GLCM). A escolha se deu por este apresentar ótimos resultados na literatura, incluindo vários trabalhos voltados a verificação de defeitos em pisos cerâmicos. Entre estes, podemos citar: Mansoory, Tajik e Pashna (2009), Sharma e Kaur (2012), Fathi, Monadjemi e Mahmoudi (2012), Chen et al. (2013) e Senthilkumar, Palanisamy e Jaya (2015).

Neste trabalho, optou-se por criar uma GLCM com 256 níveis. Ou seja, ela possui dimensões de 256×256. A distância escolhida foi 1 e um ângulo de $90°$. Isso quer dizer que os pixeis serão comparados com o seu vizinho imediato à direita. Após ela ser montada, foi normalizada para o cálculo de todos os parâmetros.

Baseado no algoritmo disponibilizado por Uppuluri (2008) e à partir da matriz montada, são calculados 23 parâmetros. Estes, estão matematicamente descritos na subseção 2.11.1.2. Depois do cálculo dos mesmos, os parâmetros são organizados em um vetor de características para a posterior classificação.

4.5.2 Utilização do SFTA para extração de características

Além do método citado anteriormente, escolheu-se por verificar como o SFTA se comporta em aplicações envolvendo pisos cerâmicos. Este método foi escolhido também por trazer bons resultados na literatura, inclusive com imagens texturizadas. Entre os principais trabalhos, podemos citar: Arivazhagan et al. (2015), El-Henawy, Bakry e Hadad (2016), Samiappan et al. (2017) e Macarini e Weber (2017)[1]. Apenas o último envolve a classificação de pisos cerâmicos.

O único parâmetro escolhido pelo usuário é o n_t. Ele representa a quantidade de vezes em que o *Multi-Level Otsu Algorithm* é aplicado recursivamente na imagem. Este parâmetro influencia no tamanho do vetor de características extraído. Esta quantidade é $2 \times 3 \times n_t$. Neste trabalho, $n_t = 8$, gerando um vetor com 48 características. Este parâmetro foi escolhido de modo experimental.

4.6 CLASSIFICAÇÃO

A etapa de classificação consiste em utilizar as caraterísticas extraídas de cada imagem para inseri-la em uma determinada classe. Assim, baseado nestes dados estatísticos, a imagem é classificada de acordo com o treinamento do algoritmo de classificação.

Diferentes classificadores tem diferentes características. Sendo assim, é difícil afirmar que existe um que se sobressai em todos os tipos de aplicação. Logo, optou-se por testar diferentes algoritmos de classificação com o intuito de obter o melhor resultado para esta aplicação.

4.6.1 Redes Neurais Artificiais

As Redes Neurais tem como sua principal vantagem o fato de, quanto maior for o tamanho da rede (a quantidade de camadas ocultas e seu tamanho), mais flexível ela se torna. Assim, o erro no conjunto de treinamento pode ser muito pequeno. Ao mesmo tempo, a rede treinada aprenderá sobre o "ruído" presente no mesmo. Logo, o erro do conjunto de teste geralmente começa a aumentar depois que o tamanho da rede alcança seu limite (OPENCV, 2014).

Quando a rede é muito grande, a fase de treinamento é maior.

[1] Artigo publicado em decorrência da realização deste trabalho de conclusão de curso.

Então, é razoável pré-processar os dados antes desta etapa. Para este fim, técnicas como o PCA (*Principal Component Analysis*) podem ser utilizadas (OPENCV, 2014).

Para a obtenção dos resultados, utilizou-se a implementação disponível no módulo de *Machine Learning* do OpenCV. Neste trabalho, foi utilizada uma rede neural do tipo *Multilayer Perceptron*. Sua estrutura possui 15 neurônios na camada oculta, sendo que este valor foi escolhido através de experimentos. Como função de ativação, optou-se por uma *Symmetrical Sigmoid*. Esta é descrita pela Equação 4.2.

$$f(x) = \beta \cdot \frac{1 - e^{-\alpha x}}{1 + e^{-\alpha x}} \tag{4.2}$$

O método de treinamento utilizado foi o *Backpropagation*. Como critério de parada para esta fase, determinou-se que a execução fosse encerrada com 1000 iterações ou quando o algoritmo atingisse um erro menor que 1×10^{-6}. O OpenCV utiliza a implementação proposta por LeCun et al. (2012).

4.6.2 *Support Vector Machine*

Outro classificador utilizado foi o *Support Vector Machine*. Ele foi escolhido por ser capaz de apresentar uma boa performance em uma quantidade razoável de problemas. Além disso, ele é muito eficiente e estável (CASAGRANDE et al., 2017). Esta técnica apresenta uma boa capacidade de generalização em situações reais, onde geralmente obtêm-se resultados melhores do que com outros classificadores, tanto em predições quando em classificações (AKANDE et al., 2014).

Para a implementação deste classificador, foi utilizada a LIBSVM (CHANG; LIN, 2011). Esta biblioteca foi escolhida por sua facilidade de uso e pelo grande volume de documentação. Além disso, pode ser facilmente integrada ao código-fonte da aplicação em desenvolvimento.

Antes de treinar ou mesmo utilizar o SVM para predição, é importante normalizar os dados. As duas principais vantagens são: evitar que os atributos com valores muito altos dominem os que possuem valores menores e evitar dificuldades matemáticas durante o cálculo (HSU et al., 2003). O processo de normalização consiste em transformar os valores numéricos para encaixá-los entre dois limites: um limite inferior e um superior. Neste trabalho, os dados foram normalizados para um intervalo [-1 1].

Inicialmente, os dois principais parâmetros do SVM são des-

conhecidos. Para encontrar o par (C, γ) que traz a melhor desempenho para a aplicação, foi utilizado o *Grid Search Algorithm* (HSU et al., 2003). Este, consiste em tentar vários pares (C, γ) e selecionar aquele que trouxer a melhor acurácia utilizando validação cruzada (*cross-validation*) (CHANG; LIN, 2011). A seleção de parâmetros é uma etapa essencial, levando em conta que a escolha errada destes faz com que a taxa de acerto do sistema seja diminuída.

Neste trabalho, utilizou-se um SVM do tipo *C-Support Vector Classification* (C-SVC) para classificação multi-classe. Este tipo de SVM permite separação imperfeita de classes, com uma penalidade C para erros. Como *kernel*, utilizou-se o tipo *radial basis function* (Equação 3.2).

4.6.3 Árvores de decisão

A principal vantagem da utilização das árvores de decisão é o fato de estas possuírem a capacidade de quebrar um problema relativamente complexo em problemas menores, entregando uma solução facilmente interpretável. Além disso, enquanto alguns classificadores testam as amostras com todas as outras presentes no conjunto de dados, as árvores fazem isso com apenas uma parte destas, reduzindo a quantidade de processamento desnecessário (SAFAVIAN; LANDGREBE, 1991).

Safavian e Landgrebe (1991) ainda afirmam que outra vantagem se dá pelo fato de, muitas vezes, alguns classificadores apresentarem um problema conhecido como *Curse of Dimensionality*. Com as árvores de decisão este problema pode ser evitado. Para isto, utiliza-se uma quantidade menor de características em cada nó interno, sem que haja excessiva degradação de performance.

Para obter os resultados utilizando as árvores de decisão, foi utilizada a implementação disponibilizada no módulo de *machine learning* do OpenCV. Esta, por sua vez, foi implementada de acordo com o trabalho proposto por Breiman et al. (1984).

Para a obtenção dos melhores resultados possíveis, os parâmetros necessários para o funcionamento do algoritmo foram escolhidos de modo prático. A Tabela 2 mostra quais são estes parâmetros, suas respectivas descrições e seus valores.

Tabela 2 – Árvore de decisão: parâmetros, descrição e valores

Parâmetro	**Descrição**	**Valor**
maxDepth	Profundidade máxima da árvore	10
minSampleCount	Quantidade de amostras necessárias em um nó para que haja separação	2
regressionAccuracy	Critério de terminação para árvores de regressão (não utilizado neste caso)	0
useSurrogates	Se *true*, as chamadas *surrogate splits* são criadas	false
maxCategories	Agrupa possíveis valores de uma variável de determinada categoria em k *maxCategories* divisões para encontrar uma divisão ótima	16
CVFolds	Quantidade de *folds* utilizados (se for utilizado *k-fold cross validation*)	0
use1SERule	Se for *true*, as "podas" serão mais severas	false
truncatePrunedTree	Se *true*, os galhos podados serão fisicamente removidos da árvore	false
setPriors	Um vetor das classes probabilidades preliminares, organizadas por valor	Mat()

4.6.4 *Random Trees*

Um dos motivos que torna a utilização das *Random Trees* (também conhecidas como *Random Forest*) facilitada é o fato de existir um único parâmetro ajustável. Este representa a quantidade de variáveis utilizadas para fazer a separação nos nós. Apesar disso, o algoritmo não é sensível a este fato. Geralmente, o valor deste parâmetro é calculado como sendo a raiz quadrada da quantidade de entradas. Limitando esta quantidade, a complexidade do algoritmo também é diminuída. Além disso, as *Random Trees* não são podadas, fazendo com que a carga computacional seja diminuída (GISLASON; BENEDIKTSSON; SVEINSSON, 2006).

Isso significa que as *Random Trees* podem lidar com dados de dimensões maiores e utilizar uma quantidade alta de árvores para formar um conjunto. Isto é combinado ao fato de que a seleção aleatória de variáveis busca minimizar a correlação entre as árvores no conjunto. Assim, a taxa de erro pode ser comparada com classificadores como o *AdaBoost*, enquanto são computacionalmente mais leves (GISLASON; BENEDIKTSSON; SVEINSSON, 2006).

Tabela 3 – *Random Trees*: parâmetros, descrição e valores

Parâmetro	Descrição	Valor
maxDepth	Profundidade máxima da árvore	10
minSampleCount	Quantidade de amostras necessárias em um nó para que haja separação	2
regressionAccuracy	Critério de terminação para árvores de regressão (não utilizado neste caso)	0.f
useSurrogates	Se *true*, as chamadas *surrogate splits* são criadas	false
maxCategories	Agrupa possíveis valores de uma variável de determinada categoria em k <= maxCategories divisões para encontrar a divisão ótima	16
priors	Um vetor das classes de probabilidades preliminares, organizadas por valor	Mat()
calculateVarImportance	Se for *true*, a importância de cada variável é calculada	false
activeVarCount	O tamanho do subconjunto de características em cada nó da árvore que é utilizado para encontrar a(s) melhor(es) separação(ões)	5

Apesar de ter sido dito que o único parâmetro ajustável é a quantidade de variáveis utilizadas para fazer a separação nos nós, a Tabela 3 mostra que existem mais. Porém, deve-se perceber que os outros parâmetros são referentes as árvores de decisão (Tabela 2).

4.6.5 *k-Nearest Neighbor*

Escolheu-se incluir o kNN neste trabalho por possuir uma estrutura relativamente simples e poder fazer classificações com pouco esforço. Além disso, o período de treino é muito pequeno, sendo muitas vezes nulo (FRIEDMAN; HASTIE; TIBSHIRANI, 2001). É conhecido por ser um método não-paramétrico. Isso significa que seu desempenho não depende da escolha de seus parâmetros (ALTMAN, 1992).

O OpenCV também disponibiliza em seu módulo de *machine learning* uma implementação do kNN. Para este trabalho, a quantidade k de vizinhos escolhido foi 3 e este valor foi encontrado de forma experimental. Além disso, o tipo de algoritmo escolhido é conhecido como Força Bruta (do inglês, *Brute Force*).

O kNN de Força Bruta é um dos métodos de classificação mais simples. É um algoritmo que consiste nos seguintes passos:

1. Calcula-se a distância entre o ponto em análise e os demais pontos;

2. Ordena-se as distância encontradas;

3. Escolhe-se os k pontos de menor distância;

4. Inicia-se o processo de "votação" dos objetos;

5. Repetem-se os passos anteriores para todos os pontos a serem analisados.

Este tipo de algoritmo pode ser muito eficiente com poucas amostras de dados. Porém, sofre de um problema conhecido como *Curse of Dimensionality*. Este se refere ao problema de encontrar uma estrutura para dados que estão incorporados em espaços dimensionais maiores. Quanto maior a quantidade de características, mais dados são necessários para preencher o espaço (FRIEDMAN; HASTIE; TIBSHIRANI, 2001).

4.7 RESULTADOS

Este trabalho tem como um dos objetivos detectar a maior quantidade de defeitos visuais em pisos cerâmicos utilizando processamento de imagens e aprendizado de máquinas. Estas áreas, por serem muito amplas, possuem inúmeros algoritmos com as mais variadas abordagens.

Sendo assim, uma abordagem pode funcionar muito bem para uma aplicação, enquanto outra pode falhar totalmente. Tendo isto em mente, optou-se por analisar e fazer experimentos combinando algoritmos de processamento de imagens e classificação.

Foram utilizados dois algoritmos de extração de características e cinco classificadores. Foram gerados resultados a partir destas 10 combinações para uma posterior comparação. Com isso, procurou-se obter a melhor solução dentre as propostas, tanto em relação ao tempo de processamento quanto a taxa de acerto.

Para encontrar os parâmetros ótimos em relação aos classificadores, foi utilizada uma espécie de otimização. Esta se assemelha a um método de força bruta, já que foram feitas várias tentativas e os parâmetros escolhidos foram aqueles que trouxeram a melhor taxa de acerto.

Todos os algoritmos foram implementados em C/C++, utilizando as bibliotecas OpenCV. A unidade de processamento foi um computador com um processador Intel Core i3-4030U CPU @ 1.9GHz $\times$ 2, 8GB de memória RAM, SSD Sandisk PLUS com 240GB e uma GPU Intel Corporation Haswell - ULT Graphics Controller. O sistema

Tabela 4 – Tempos obtidos na etapa de pré-processamento

Conjunto	**Média (s)**	**Desvio Padrão (s)**
1	0,178	0,023
2	0,169	0,027
3	0,177	0,022
4	0,156	0,018
5	0,161	0,021

operacional utilizado foi o Linux Mint 18.1, *Cinnamon Edition*, 64-bit com um *kernel* Linux 4.4.0-53-*generic*.

4.7.1 Tempo de Processamento

Uma linha de produção de pisos cerâmicos tem uma alta taxa de saída. Sendo assim, a verificação de defeitos precisa funcionar em tempo real. Porém, devido ao custo, esta etapa ainda é realizada por seres humanos e este fato faz com que haja limitação na taxa de produção.

Ao propor esta abordagem, teve-se como intuito a classificação de pisos de qualquer tamanho. Para isso, o tempo de pré-processamento, processamento e classificação somados precisam ficar abaixo de 1,5 segundos por peça. Sendo assim, a escolha dos algoritmos precisa ser feita de maneira cuidadosa.

4.7.1.1 Etapa de Pré-Processamento

A etapa de pré-processamento consiste em ajustar a imagem capturada com o intuito de tentar melhorar o resultado que será obtido na etapa subsequente. Sendo assim, existe a necessidade de remover toda a informação desnecessária visando diminuir o tempo de processamento no momento da extração de características.

Além disso, tem-se o objetivo de corrigir os problemas que venham a acontecer no momento da aquisição da imagem. O caso mais comum é a angulação no piso em análise. Se o piso estiver torto no momento em que a imagem for adquirida, é necessário ajustá-lo de modo a diminuir os problemas na fase de extração de características. A Tabela 4 mostra a média e o desvio padrão dos tempos referentes a etapa de pré-processamento.

A média de tempo obtida em todos os conjuntos pode ser con-

Tabela 5 – Tempos de processamento obtidos

Conjunto	SFTA		GLCM	
	Média (s)	Desvio Padrão (s)	Média (s)	Desvio Padrão (s)
1	0,517	0,0340	0,0504	0,0103
2	0,510	0,0202	0,0461	0,0029
3	0,494	0,0152	0,0473	0,0019
4	0,604	0,0919	0,0423	0,0012
5	0,544	0,0819	0,0452	0,0022

siderada satisfatória se comparada a outros trabalhos presentes na literatura. Estes, não apresentam uma etapa de pré-processamento propriamente dita. Logo, para comparações, será utilizado o tempo total de processamento obtido. A Tabela 6 mostra estes valores.

O principal ponto positivo se deve ao fato de o desvio padrão obtido ser um valor pequeno. Isso mostra a estabilidade do algoritmo e o seu comportamento no processamento de diferentes padrões de pisos, com e sem texturas.

4.7.1.2 Etapa de Extração de Características

A etapa de extração de características consiste em utilizar os algoritmos anteriormente citados visando extrair os dados estatísticos que descrevem o piso em análise. Para isso, dois métodos de análise de textura foram empregados. Assim, os tempos de processamento e as taxas de acerto poderiam ser comparadas.

Para a obtenção dos resultados, mediu-se o tempo de processamento para cada piso individualmente e então, calculou-se a média e o desvio padrão dos tempos obtidos em cada padrão de piso presente na base de dados. Os resultados podem ser vistos na Tabela 5.

Primeiramente, é preciso enfatizar que o tempo despendido na etapa de classificação não foi incluído na análise por ser muito baixo. Este, foi observado durante a fase de obtenção de resultados e teve sua média na ordem de 10^{-2} segundos.

Pode-se perceber pelas Tabelas 4 e 5 que o tempo necessário para processar as imagens dos pisos fica abaixo de um segundo. Esse fato faz com fique claro que ambos os algoritmos funcionam de maneira satisfatória em relação a este quesito.

Outro fato importante a ser destacado é que dentre os padrões utilizados na análise, existem dois com textura e três destes sem. Ainda assim, o tempo despendido nas duas etapas (pré-processamento e pro-

Tabela 6 – Comparativo - Média de Tempo de Execução (Por piso)

Trabalho	**Tempo (s)**
Este trabalho (GLCM)	0,21
Este trabalho (SFTA)	0,70
Meena e Mittal (2013)	0,44
Mishra e Shukla (2014)	0,18
Mohan e Kumar (2015)	0,18
Hocenski, Matić e Vidović (2016)	0,90
Hanzaei, Afshar e Barazandeh (2017)	3,52

cessamento) se manteve abaixo do encontrado na indústria.

Os resultados obtidos poderiam ser melhorados se fosse utilizado um *hardware* com mais poder de processamento. Utilizar GPUs com suporte a CUDA (*Compute Unified Device Architecture*) diminuiria significativamente o tempo de processamento despendido em cada imagem. Isto porque o OpenCV fornece suporte a esta API, permitindo o processamento paralelo.

É importante citar que os tempos referentes a este trabalho (Tabela 6) foram obtidos calculando a média de tempo dos cinco conjuntos de imagens utilizados. Além disso, é resultado da soma do tempo despendido nas etapas de pré-processamento e extração de características. O tempo médio necessário para a classificação dos pisos foi desconsiderado por ser um valor muito pequeno se comparado aos atingidos em outras etapas do sistema.

4.7.2 Taxa de acerto

A taxa de acerto é um parâmetro muito importante a ser levado em conta neste tipo de aplicação. Isto porque em uma indústria, sendo ela de qualquer segmento, a qualidade dos produtos é um ponto imprescindível e é levado em conta em qualquer decisão feita pelas empresas. Sendo assim, a taxa de acerto obtida através desta abordagem precisa ser alta.

Para a obtenção dos resultados, os conjuntos de imagens foram divididos na proporção 80/20. Ou seja, 80% das imagens foram utilizadas para treinamento dos algoritmos de classificação e 20% foram usadas para testes. A Tabela 7 mostra como foram divididos cada conjunto.

As imagens foram separadas de modo aleatório da seguinte ma-

Tabela 7 – Divisão dos conjuntos de imagens em treinamento e teste

Conjunto	**Treino**	**Teste**
1	130	32
2	144	36
3	132	33
4	322	81
5	303	76

Tabela 8 – Taxa de acerto - Rede Neural

Conjunto	**SFTA**	**GLCM**
1	90,63%	96,88%
2	86,11%	97,22%
3	90,91%	87,88%
4	88,89%	86,42%
5	69,74%	96,05%

neira: dentre as imagens que representam os pisos sem defeitos, 80% destas foram selecionadas e utilizadas para treinamento. O restante, foi utilizada para teste. As imagens que representam os pisos com defeitos foram selecionadas da mesma maneira. É importante destacar que aquelas escolhidas para teste são totalmente desconhecidas pelo classificador.

Uma das principais vantagens das redes neurais artificiais é o fato de, quanto maior é a sua camada oculta, mais flexível ela se torna. Porém, é necessário levar em conta a taxa de acerto necessária e o tempo máximo permitido para treinamento. Isso acontece pois quanto maior é a rede, mais tempo ela leva para treinar. Então deve haver um meio termo entre as duas variáveis. Os resultados relacionados a taxa de acerto são apresentados na Tabela 8.

As árvores de decisão tem como uma de suas vantagens a classificação de modo preciso, mesmo quando há uma quantidade relativamente baixa de amostras para treinamento. Sendo assim, a Tabela 9 mostra os resultados obtidos utilizando este classificador.

O principal benefício da utilização das *random trees* é o fato deste classificador não necessitar de ajuste de parâmetros. Ou seja, é um método não-paramétrico. Na Tabela 10 são apresentados os dados referentes as taxas de acerto atingidas utilizando este método de classificação.

Um dos principais pontos positivos da utilização do *k-nearest*

Tabela 9 – Taxa de acerto - Árvores de Decisão

Conjunto	SFTA	GLCM
1	93,75%	93,75%
2	80,56%	94,44%
3	93,94%	90,91%
4	98,76%	98,76%
5	86,84%	94,74%

Tabela 10 – Taxa de acerto - *Random Trees*

Conjunto	SFTA	GLCM
1	96,87%	93,75%
2	88,89%	97,22%
3	90,91%	93,94%
4	96,30%	95,06%
5	96,05%	97,37%

neighbor é o fato deste possuir um período de treinamento relativamente pequeno se comparado a outros classificadores. Isso faz com que sua aplicação seja recomendável em sistemas de tempo real. Na Tabela 11 são apresentados os resultados obtidos com a utilização deste algoritmo.

O *Support Vector Machine* tem como principal ponto positivo o fato deste poder lidar com dados lineares e não-lineares. Além disso, a variedade de *kernels* que podem ser utilizados faz com que a quantidade de aplicações em que este classificador possa ser usado se torne ainda maior. Assim sendo, as taxas de acerto obtidas utilizando o SVM são apresentadas na Tabela 12.

Utilizando os dados obtidos na etapa de classificação, além da taxa de acerto (Equação 4.3), podem-se calcular outros importantes

Tabela 11 – Taxa de acerto - *k-Nearest Neighbor*

Conjunto	SFTA	GLCM
1	90,63%	93,75%
2	88,89%	69,44%
3	93,94%	84,85%
4	96,26%	90,12%
5	96,05%	89,47%

Tabela 12 – Taxa de acerto - *Support Vector Machine*

Conjunto	SFTA	GLCM
1	81,35%	96,87%
2	83,33%	94,44%
3	87,88%	93,94%
4	97,53%	98,76%
5	98,68%	98,68%

parâmetros. O primeiro, é conhecido como Sensitividade (Equação 4.4) e representa a taxa de *true positive*. Ou seja, descreve a taxa das amostras que foi classificada de modo correto (MACARINI; WEBER, 2017).

Outro importante parâmetro é a Especificidade (Equação 4.5). Ao contrário do anterior, este é conhecido como o *true negative rate*, pois descreve a probabilidade em que o sistema classifica um piso sem defeitos como "bom" (MACARINI; WEBER, 2017).

$$Taxa\ de\ Acerto = \frac{TP + TN}{TP + TN + FP + FN} \times 100 \tag{4.3}$$

$$Sensitividade = \frac{TP}{TP + FN} \times 100 \tag{4.4}$$

$$Especificidade = \frac{TN}{TN + FP} \times 100 \tag{4.5}$$

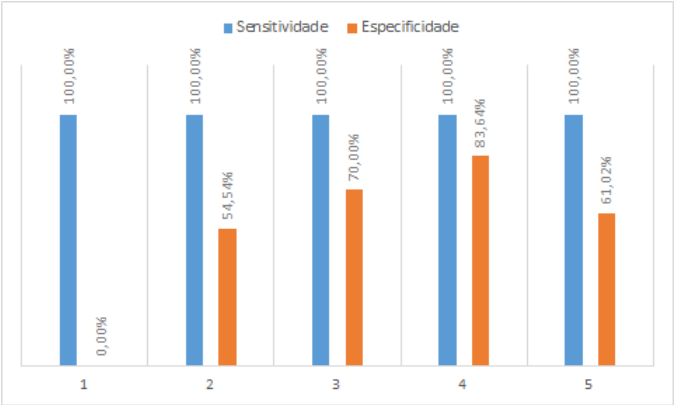
As abreviaturas utilizadas nas equações acima são definidas como:

- TN: *True Negative*. Representa a quantidade de pisos que não possuem nenhum defeito e o sistema classifica eles de modo correto;
- TP: *True Positive*. Descreve o número de pisos que possuem defeitos e o sistema os classifica como defeituosos;
- FN: *False Negative*. Representa a taxa de pisos que são classificados como "bons", mas que possuem defeitos;
- FP: *False Positive*. É a quantidade de pisos que não possuem nenhum defeito, mas o sistema os classifica como "ruins".

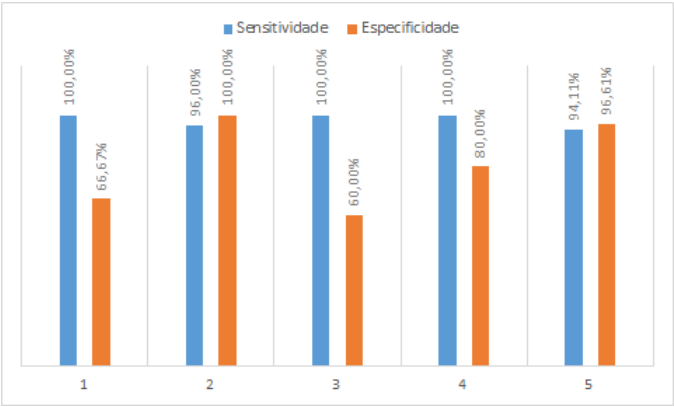
Os resultados obtidos em relação a sensitividade e a especificidade utilizando redes neurais são mostrados na Figura 23. Apesar de as

taxas de sensitividade terem sido boas, as relacionadas a especificidade não foram. Isso se deu principalmente no três primeiros conjuntos de imagens, onde o desequilíbrio entre a quantidade de pisos bons e ruins é mais acentuada. O pior caso ocorreu quando o SFTA foi utilizado para extração de características no primeiro conjunto de dados, onde foi atingido um valor de 0%.

Figura 23 – Resultados obtidos utilizando Redes Neurais



(a) *SFTA*



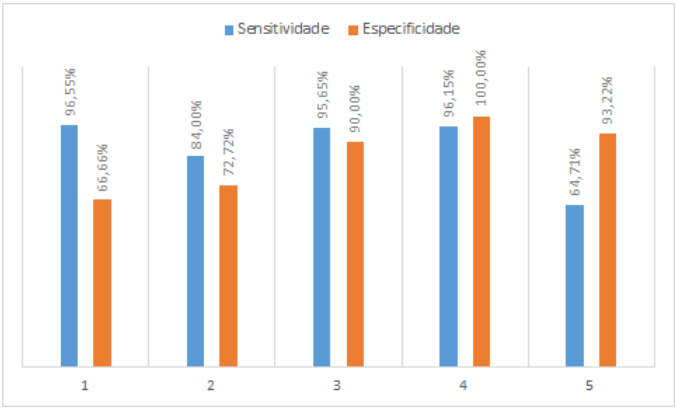
(b) GLCM

Fonte: Elaborada pelo autor

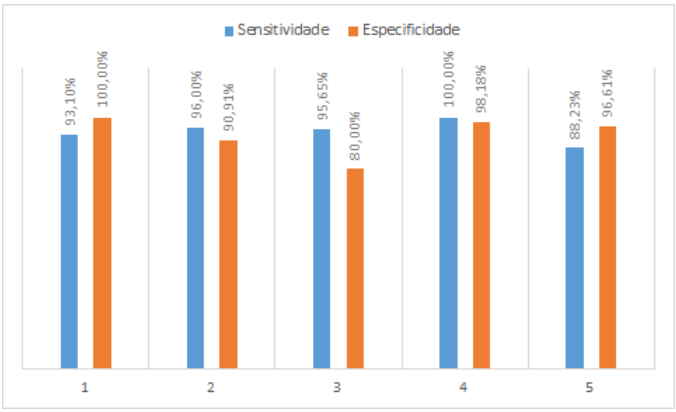
A Figura 24 mostra estes parâmetros calculados utilizando os resultados obtidos através do uso de árvores de decisão. Pode-se per-

ceber que houve certa variação nestes valores. Isso acontece devido ao desbalanceamento do conjunto de dados. Este fato influencia de modo direto nos resultados obtidos.

Figura 24 – Resultados obtidos utilizando Árvores de Decisão



(a) *SFTA*

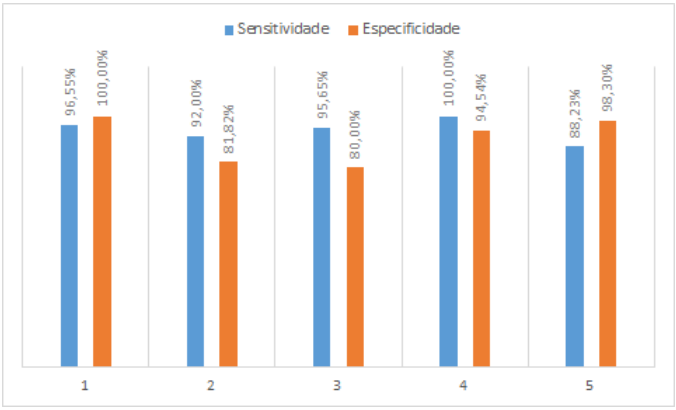


(b) GLCM

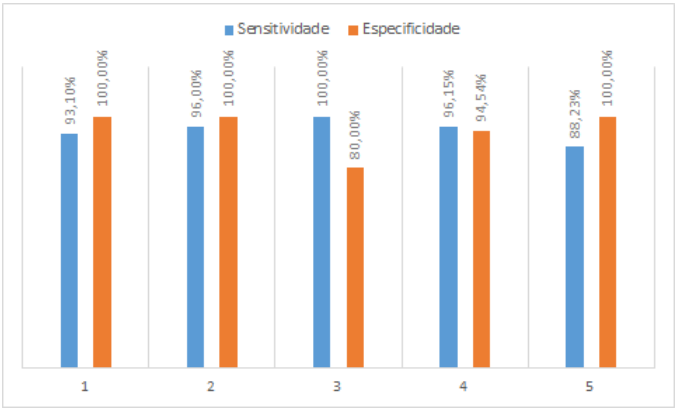
Fonte: Elaborada pelo autor

Utilizando as *random trees*, as taxas de especificidade e sensitividade tiveram uma variação menor. Isso é um ponto positivo para a estabilidade do sistema, já que mesmo com o conjunto de dados razoavelmente desbalanceado, o classificador conseguiu manter uma taxa de acerto aceitável.

Figura 25 – Resultados obtidos utilizando as *Random Trees*



(a) *SFTA*



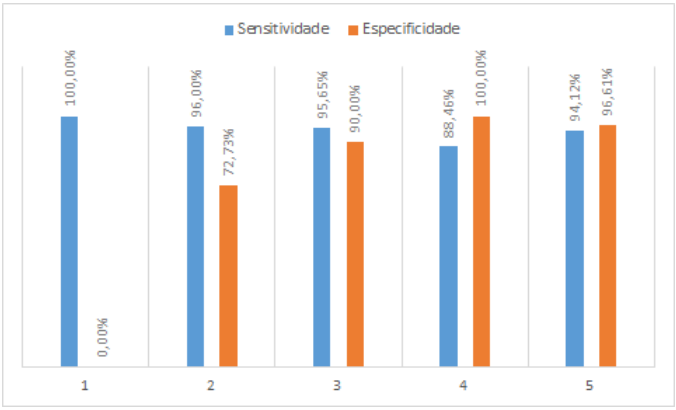
(b) GLCM

Fonte: Elaborada pelo autor

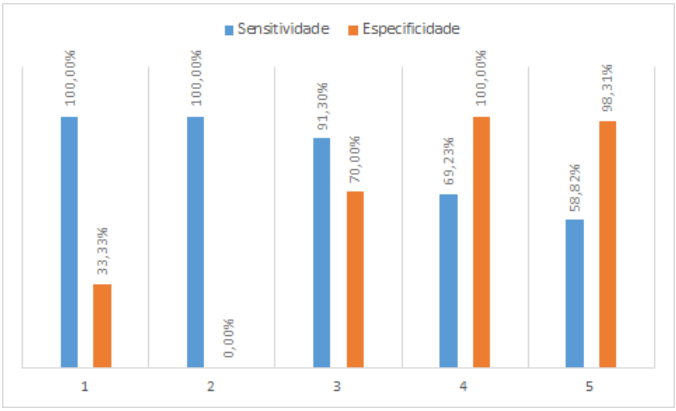
Utilizando o kNN, o cenário foi diferente. Pode-se observar na Figura 26(a) que a especificidade para o primeiro conjunto de dados foi 0%. Apesar de este ser um fato preocupante, neste caso, o sistema classificaria um piso bom como ruim. Como dito anteriormente, apesar de ser um fato negativo, se o contrário acontecesse seria pior. Ou seja, se um piso ruim fosse classificado como bom, comprometeria a qualidade da linha de produção.

Além disso, o kNN mostrou taxas relativamente baixas de espe-

Figura 26 – Resultados obtidos utilizando kNN



(a) *SFTA*



(b) GLCM

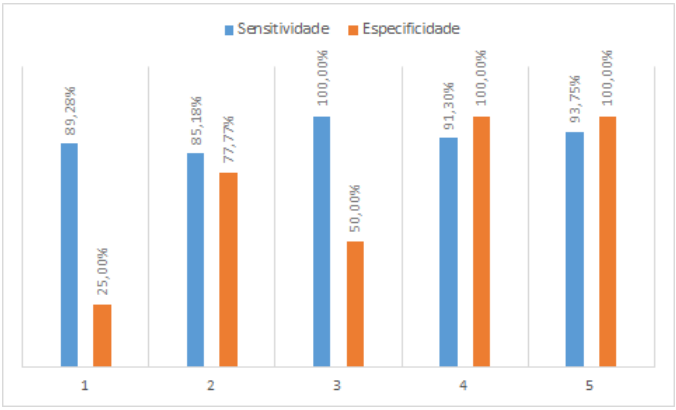
Fonte: Elaborada pelo autor

cificidade para os três primeiros conjuntos de imagens, quando utilizado o GLCM. Isso acontece porque estes possuem o desbalanceamento mais acentuado se comparado aos outros conjuntos de imagens.

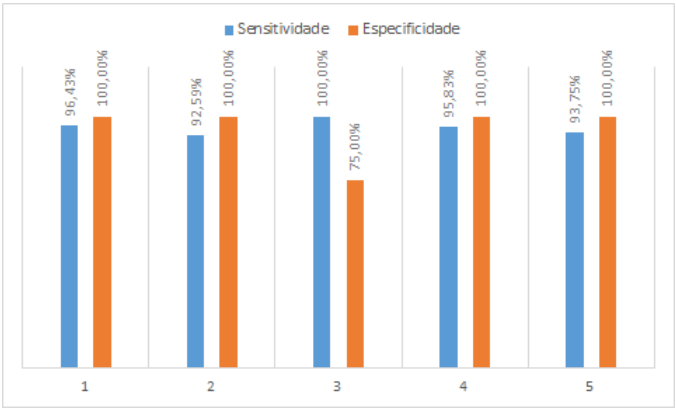
Em relação ao *Support Vector Machine*, utilizando o SFTA como algoritmo de extração, os resultados apresentados são similares ao do kNN. Isso mostra que o SVM também não se comporta tão bem para cenários onde o desequilíbrio entre os dados é grande. Com a utilização do GLCM, os resultados mostraram uma variação menor, tornando o

sistema mais estável.

Figura 27 – Resultados obtidos utilizando SVM



(a) *SFTA*



(b) GLCM

Fonte: Elaborada pelo autor

Em relação ao tempo de processamento, os resultados atingidos com métodos utilizados foram bons. Em todos os cenários, o tempo de processamento fica abaixo de um segundo, atingindo um tempo menor do que o presente atualmente na indústria. Isso mostra a eficiência da abordagem proposta, tanto para pisos que possuem textura quanto para os que não têm.

O SFTA mostrou um desempenho superior na extração de carac-

terísticas em pisos que possuem textura se comparados àqueles que não têm. Ainda assim, o desempenho do GLCM foi extremamente superior, sendo 10 vezes mais rápido que o primeiro algoritmo, se considerado o pior caso.

Analisando a taxa de acerto, pode-se perceber que dois classificadores se destacaram: as *random forests* e o *support vector machine*. O primeiro obteve a melhor média em relação a taxa de acerto quando utilizou-se o SFTA como algoritmo de extração. O classificador atingiu 93,80% de média, levando em conta os cinco conjuntos de imagens. Além disso, o mesmo obteve seu valor máximo utilizando o primeiro conjunto de imagens, onde houve 96,87% de acerto.

Já quando o GLCM foi utilizado como algoritmo de extração, o SVM obteve a melhor média em relação a taxa de acerto, atingindo 96,54%. Utilizando o quinto conjunto de imagens, o classificador alcançou 98,68% de acerto, sendo este o maior valor de acurácia deste trabalho.

5 CONCLUSÃO

O interesse nos algoritmos de processamento de imagens se dá por dois motivos principais. O primeiro é melhorar a informação contida nas imagens para a interpretação humana. O segundo é processar a imagem para o armazenamento, transmissão ou para que ela seja posteriormente interpretada por uma máquina.

Processamento de imagens é um tópico bastante abordado na literatura. Suas técnicas são utilizadas em áreas como reconhecimento de padrões, médica e na manipulação de imagens e vídeos. Além disso, a verificação de defeitos vem sendo bastante explorada. Mais especificamente, a verificação de defeitos em pisos cerâmicos possui uma vasta literatura.

O aprendizado de máquina tem como objetivo gerar informação à partir de dados. Isto é feito extraindo regras ou padrões dos mesmos. Depois de aprender utilizando uma coleção de dados de treinamento, espera-se que a máquina seja capaz de generalizar modo preciso. Este aprendizado pode ocorrer de modo supervisionado ou não-supervisionado.

Este trabalho propôs uma abordagem para verificação de defeitos visuais em pisos cerâmicos. Esta solução foi fundamentada em processamento de imagens e aprendizado de máquina. Em busca da melhor solução, testou-se dois algoritmos de extração e cinco classificadores. A solução foi implementada em C++ utilizando as bibliotecas OpenCV.

Os resultados obtidos foram satisfatórios, tanto em relação ao tempo de processamento quanto a taxa de acerto. As taxas de acerto médias foram de 93,80% e 96,54%. Eles podem ser comparados a outros trabalhos presentes na literatura, incluindo os citados neste. Porém, em alguns deles os métodos de extração e classificação utilizados não foram informados pelos autores. Isso faz com que seja difícil comparar os resultados obtidos neste trabalho com os propostos na literatura.

Sabe-se que uma câmera de melhor qualidade poderia aumentar a taxa de acerto do sistema. Porém, é necessário levar em conta que a proposta de manter o projeto com baixo custo foi mantida. Além disso, um *hardware* com mais poder de processamento poderia ser utilizado, fazendo com que o tempo de processamento das imagens fosse diminuído. Um exemplo seria utilizar GPUs que tenham suporte a CUDA. As bibliotecas do OpenCV oferecem soluções que tiram proveito deste recurso, se utilizando de processamento paralelo.

O principal ponto negativo deste trabalho é a simplicidade do

sistema de aquisição de imagens. Porém, como já foi citado anteriormente, teve-se como objetivo manter o sistema com baixo custo.

Apesar disso, os resultados mostram o potencial da abordagem, que foi capaz de classificar pisos com e sem textura. Este fato mostra que ela pode ser utilizada em uma linha de produção. Isso se dá principalmente pelo tempo de processamento atingido, permitindo que o sistema faça a classificação em menos de um segundo.

Como resultado deste trabalho de conclusão de curso, foram publicados dois artigos. O primeiro, Casagrande et al. (2017), foi publicado no Simpósio Brasileiro de Automação Inteligente (SBAI), organizado pela Universidade Federal do Rio Grande do Sul (UFRGS) e Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS). O segundo, Macarini e Weber (2017), foi submetido ao *30^{th} Conference on Graphics, Patterns and Images - SIBGRAPI*, evento patrocinado pela Sociedade Brasileira de Computação (SBC).

Após os fatos apontados, acredita-se que o objetivo geral deste trabalho foi alcançado. As contribuições presentes neste podem ser utilizadas como base para o desenvolvimento de outras soluções, voltadas a diversas áreas. Uma solução para a indústria pode ser implementada utilizando esta abordagem, desde que sejam aplicadas as melhorias necessárias.

5.1 PROPOSTAS PARA TRABALHOS FUTUROS

- Classificar os tipos de defeitos encontrados nas imagens;
- Otimizar os algoritmos utilizando GPUs com suporte a CUDA;
- Portar e otimizar o projeto para que ele possa funcionar em um sistema embarcado ou um *single board computer*, como o *Raspberry Pi*;
- Fazer melhorias no sistema de aquisição de imagens, visando o aumento de desempenho do sistema;
- Verificar o funcionamento da abordagem em uma linha de produção real;
- Implementar outros algoritmos de extração e classificação para realizar mais comparações;
- Utilizar algoritmos de busca/otimização (algoritmo genético, por exemplo) visando obter melhores resultados.

REFERÊNCIAS

AKANDE, K. O. et al. Performance comparison of svm and ann in predicting compressive strength of concrete. **IOSR Journal of Computer Engineering**, v. 16, n. 5, p. 88–94, 2014.

ALTMAN, N. S. An introduction to kernel and nearest-neighbor nonparametric regression. **The American Statistician**, Taylor & Francis, v. 46, n. 3, p. 175–185, 1992.

ARIVAZHAGAN, S. et al. Railway track derailment inspection system using segmentation based fractal texture analysis. **ICTACT Journal on Image & Video Processing**, v. 6, n. 1, 2015.

ARTHUR, D.; VASSILVITSKII, S. k-means++: The advantages of careful seeding. In: SOCIETY FOR INDUSTRIAL AND APPLIED MATHEMATICS. **Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms**. [S.l.], 2007. p. 1027–1035.

BASU, A.; LI, X. **Computer Vision: Systems, Theory and Applications**. [S.l.]: World Scientific Publishing Co. Inc., 1993. ISBN 981-02-1392-1.

BASU, M. Gaussian-based edge-detection methods - a survey. **IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)**, IEEE, v. 32, n. 3, p. 252–260, 2002.

BHARATI, M. H.; LIU, J. J.; MACGREGOR, J. F. Image texture analysis: Methods and comparisons. **Chemometrics and intelligent laboratory systems**, Elsevier, v. 72, n. 1, p. 57–71, 2004.

BISWAS, G. et al. Cellular architecture for affine transforms on raster images. **IEE Proceedings-Computers and Digital Techniques**, IET, v. 143, n. 2, p. 103–110, 1996.

BRADSKI, G.; KAEHLER, A. **Learning OpenCV: Computer vision with the OpenCV library**. [S.l.]: O'Reilly Media, Inc., 2008.

BREIMAN, L.; CUTLER, A. **Random Forests**. 2003. Disponível em: <http://www.stat.berkeley.edu/ breiman/RandomForests/>.

BREIMAN, L. et al. **Classification and regression trees**. [S.l.]: CRC press, 1984.

CASAGRANDE, L. et al. Sistema de controle de qualidade visual de pisos cerâmicos fundamentado em processamento de imagem e aprendizado de máquina. In: **Simpósio Brasileiro de Automação Inteligente**. [S.l.: s.n.], 2017.

CHANG, C.-C.; LIN, C.-J. Libsvm: A library for support vector machines. **ACM Trans. Intell. Syst. Technol.**, ACM, New York, NY, USA, v. 2, n. 3, p. 27:1–27:27, maio 2011. ISSN 2157-6904. Disponível em: <http://doi.acm.org/10.1145/1961189.1961199>.

CHEN, S. et al. Automated inspection of engineering ceramic grinding surface damage based on image recognition. **The International Journal of Advanced Manufacturing Technology**, Springer, v. 66, n. 1-4, p. 431–443, 2013.

CLAUSI, D. A. An analysis of co-occurrence texture statistics as a function of grey level quantization. **Canadian Journal of remote sensing**, NRC Research Press Ottawa, Canada, v. 28, n. 1, p. 45–62, 2002.

COPPIN, B. **Inteligência Artificial**. [S.l.]: Grupo Editorial Nacional (GEN), 2013. ISBN 978-85-216-1729-7.

COSTA, A. F.; HUMPIRE-MAMANI, G.; TRAINA, A. J. M. An efficient algorithm for fractal analysis of textures. In: IEEE. **Graphics, Patterns and Images (SIBGRAPI), 2012 25th SIBGRAPI Conference on**. [S.l.], 2012. p. 39–46.

DAVIES, E. R. **Computer and Machine Vision: Theory, Algorithms, Practicalities**. 4. ed. [S.l.]: Elsevier Science Publishing Co Inc, 2012. ISBN 978-0-12-386908-1.

EL-HENAWY, I.; BAKRY, H. M. E.; HADAD, H. M. E. Cattle identification using segmentation-based fractal texture analysis and artificial neural networks. **Int. J. Electron. Inf. Eng**, v. 4, n. 2, p. 82–93, 2016.

ELBEHIERY, H.; HEFNAWY, A.; ELEWA, M. Surface defects detection for ceramic tiles using image processing and morphological techniques. **World Academy of Science, Engineering and Technology International Journal of Computer, Electrical,**

Automation, Control and Information Engineering Vol:1, No:5, 2007, p. 52, 2005.

ELEYAN, A.; DEMIREL, H. Co-occurrence matrix and its statistical features as a new approach for face recognition. **Turkish Journal of Electrical Engineering & Computer Sciences**, The Scientific and Technological Research Council of Turkey, v. 19, n. 1, p. 97–107, 2011.

FACELI, K. et al. **Inteligência Artificial: Uma Abordagem de Aprendizado de Máquina**. [S.l.]: Grupo Editorial Nacional (GEN), 2011. ISBN 978-85-216-1880-5.

FATHI, A.; MONADJEMI, A. H.; MAHMOUDI, F. Defect detection of tiles with combined undecimated wavelet transform and glcm. **International Journal of Soft Computing and Engineering (IJSCE)**, Citeseer, 2012.

FRIEDL, M. A.; BRODLEY, C. E. Decision tree classification of land cover from remotely sensed data. **Remote sensing of environment**, Elsevier, v. 61, n. 3, p. 399–409, 1997.

FRIEDMAN, J.; HASTIE, T.; TIBSHIRANI, R. **The elements of statistical learning**. [S.l.]: Springer series in statistics New York, 2001.

GARDNER, M. W.; DORLING, S. Artificial neural networks (the multilayer perceptron) - a review of applications in the atmospheric sciences. **Atmospheric environment**, Elsevier, v. 32, n. 14, p. 2627–2636, 1998.

GISLASON, P. O.; BENEDIKTSSON, J. A.; SVEINSSON, J. R. Random forests for land cover classification. **Pattern Recognition Letters**, Elsevier, v. 27, n. 4, p. 294–300, 2006.

GOMES, A. C. et al. Rotas estratégicas setoriais para a indústria catarinense 2022: Cerâmica. **Programa de Desenvolvimento Industrial Catarinense (PDIC)**, 2014. Acesso em: 19/03/2017. Disponível em: <http://www4.fiescnet.com.br/images/home-pedic/Ceramica_Caderno.pdf>.

GONZALEZ, R. C.; WOODS, R. E. **Processamento Digital de Imagens**. 3. ed. [S.l.]: Pearson Prentice Hall, 2010. ISBN 978-85-7605-401-6.

HANZAEI, S. H.; AFSHAR, A.; BARAZANDEH, F. Automatic detection and classification of the ceramic tiles' surface defects. **Pattern Recognition**, Elsevier, v. 66, p. 174–189, 2017.

HARALICK, R. M.; SHANMUGAM, K.; DINSTEIN, I. Textural features for image classification. **IEEE Transactions on systems, man, and cybernetics**, Ieee, v. 3, n. 6, p. 610–621, 1973.

HECHT-NIELSEN, R. et al. Theory of the backpropagation neural network. **Neural Networks**, v. 1, n. Supplement-1, p. 445–448, 1988.

HIGGINS, G. C.; WOLFE, R. N. The relation of definition to sharpness and resolving power in a photographic system. **JOSA**, Optical Society of America, v. 45, n. 2, p. 121–129, 1955.

HOCENSKI, Ž.; MATIĆ, T.; VIDOVIĆ, I. Technology transfer of computer vision defect detection to ceramic tiles industry. In: IEEE. **Smart Systems and Technologies (SST), International Conference on**. [S.l.], 2016. p. 301–305.

HONEYCUTT, C. E.; PLOTNICK, R. Image analysis techniques and gray-level co-occurrence matrices (glcm) for calculating bioturbation indices and characterizing biogenic sedimentary structures. **Computers & Geosciences**, Elsevier, v. 34, n. 11, p. 1461–1472, 2008.

HSU, C.-W. et al. A practical guide to support vector classification. 2003.

ISLAM, M. M.; SAHRIAR, M. R. An enhanced automatic surface and structural flaw inspection and categorization using image processing both for flat and textured ceramic tiles. **International Journal of Computer Applications**, Foundation of Computer Science, v. 48, n. 3, 2012.

JACOB, G.; SHENBAGAVALLI, R.; KARTHIKA, S. Detection of surface defects on ceramic tiles based on morphological techniques. **arXiv.org - Cornell University Library**, 2016.

JÄHNE, B. **Digital Image Processing**. [S.l.]: Springer, 2002. ISBN 3-540-67754-2.

JAIN, R.; KASTURI, R.; SCHUNCK, B. G. **Machine Vision**. [S.l.]: McGraw-Hill Science/Engineering/Math, 1995. ISBN 0-07-032018-7.

KARIMI, M. H.; ASEMANI, D. Surface defect detection in tiling industries using digital image processing methods: Analysis and evaluation. **ISA transactions**, Elsevier, v. 53, n. 3, p. 834–844, 2014.

LECUN, Y. A. et al. Efficient backprop. In: **Neural networks: Tricks of the trade**. [S.l.]: Springer, 2012. p. 9–48.

LIAO, P.-S. et al. A fast algorithm for multilevel thresholding. **J. Inf. Sci. Eng.**, v. 17, n. 5, p. 713–727, 2001.

LIN, S.-W. et al. Particle swarm optimization for parameter determination and feature selection of support vector machines. **Expert systems with applications**, Elsevier, v. 35, n. 4, p. 1817–1824, 2008.

MACARINI, L. A.; WEBER, T. O. Quality control system for ceramic tiles using segmentation-based fractal texture analysis and svm. In: TORCHELSEN, R. P. et al. (Ed.). **Proceedings...** 2017. Disponível em: <http://urlib.net/sid.inpe.br/sibgrapi/2017/08.29.01.15>. Acesso em: 2017, Oct. 21.

MALIK, S.; KUMAR, T. Comparative analysis of edge detection between gray scale and color image. **Communication on Applied Electronics**, v. 5, n. 2, p. 38–43, 2016.

MANSOORY, M. S.; TAJIK, H.; PASHNA, M. Surface defect isolation in ceramic tile based on texture feature analysis using radon transform and fcm. In: IEEE. **2009 International Conference on Signal Processing Systems**. [S.l.], 2009. p. 85–90.

MARR, D. **Vision**. [S.l.]: W. H. Freeman and Company, New York, 1982.

MATHWORKS. **Image Types in the Toolbox**. 2017. Online. Acesso em: 03/04/2017. Disponível em: <https://www.mathworks.com/help/images/image-types-in-the-toolbox.html>.

MATLAB, M. **Affine Transformation**. 2017. Disponível em: <https://www.mathworks.com/discovery/affine-transformation.html>.

MEENA, Y.; MITTAL, D. A. Blobs & crack detection on plain ceramic tile surface. **International Journal of Advanced**

Research in Computer Science & Software Engineering, v. 3, n. 7, 2013.

MISHRA, R.; SHUKLA, D. An automated ceramic tiles defect detection and classification system based on artificial neural network. **Int. J. Emerg. Technol. Adv. Eng**, Citeseer, v. 4, n. 3, 2014.

MITCHELL, T. M. **Machine Learning**. [S.l.]: McGraw-Hill Science/Engineering/Math, 1997. ISBN 0070428077.

MOHAN, V.; KUMAR, S. S. An automated tiles defect detection. **International Journal of Computer Applications**, Foundation of Computer Science, v. 109, n. 11, 2015.

NETO, M. S. **GSD, o que é isso?** 2016. Online. Acesso em: 12/10/2017. Disponível em: <http://blog.droneng.com.br/gsd/>.

NIELSEN, M. A. **Neural networks and deep learning**. [S.l.]: Determination Press USA, 2015.

OpenCV. **Clustering**. 2014. Disponível em: <http://docs.opencv.org/3.0-beta/modules/core/doc/clustering.html>.

OPENCV. **Neural Network**. 2014. Online. Acesso em: 24/09/2017. Disponível em: <http://docs.opencv.org/2.4/modules/ml/doc/neural_networks.html>.

OpenCV. **Understanding k-Nearest Neighbor**. 2014. Disponível em: <http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_ml/py_knn/py_knn_understanding/py_knn_understanding.ht

OpenCV. **Affine Transformations**. 2015. Disponível em: <http://docs.opencv.org/3.1.0/d4/d61/tutorial_warp_affine.html>.

PEDRINI, H.; SCHWARTZ, W. R. **Análise de imagens digitais: princípios, algoritmos e aplicações**. [S.l.]: Thomson Learning, 2008.

PITAS, I. **Digital Image Processing Algorithms and Applications**. [S.l.]: John Wiley & Sons Inc., 2000. ISBN 0-471-37739-2.

PIX4D. **TOOLS - GSD Calculator**. 2017. Online. Acesso em: 12/10/2017. Disponível em: <https://support.pix4d.com/hc/en-us/articles/202560249-TOOLS-GSD-Calculator>.

REED, J. S. **Principles of ceramics processing**. [S.l.]: Wiley New York, 1995. ISBN 0-471-59721-X.

ROSENBERG, C. **The Rest of Lenna Story**. 2001. Online. Acesso em: 03/04/2017. Disponível em: <http://www.cs.cmu.edu/~chuck/lennapg>.

RUSSELL, S.; NORVIG, P. **Inteligência Artificial**. 2. ed. [S.l.]: Elsevier, 2004. ISBN 978-85-352-1177-1.

SAFAVIAN, S. R.; LANDGREBE, D. A survey of decision tree classifier methodology. **IEEE transactions on systems, man, and cybernetics**, IEEE, v. 21, n. 3, p. 660–674, 1991.

SAMIAPPAN, S. et al. Using unmanned aerial vehicles for high-resolution remote sensing to map invasive phragmites australis in coastal wetlands. **International Journal of Remote Sensing**, Taylor & Francis, v. 38, n. 8-10, p. 2199–2217, 2017.

SANDBERG, K. Introduction to image processing in matlab. **Department of Applied Mathematics, University of Colorado at Boulder**, 2000.

SENTHILKUMAR, M.; PALANISAMY, V.; JAYA, J. An efficient feature fusion technique for surface grading of ceramic tiles. **Middle-East Journal of Scientific Research**, v. 23, n. 1, p. 59–65, 2015.

SHARIFI, M.; FATHY, M.; MAHMOUDI, M. T. A classified and comparative study of edge detection algorithms. In: IEEE. **Proceedings of the International Conference on Information Technology: Coding and Computing**. [S.l.], 2002. p. 117–120.

SHARMA, M.; KAUR, G. Integrated approach for defect detection in ceramic tiles. **International Journal Of Computer & Technology**, Citeseer, v. 3, n. 2, 2012.

SHRIVAKSHAN, G.; CHANDRASEKAR, C. A comparison of various edge detection techniques used in image processing. **IJCSI International Journal of Computer Science Issues**, Citeseer, v. 9, n. 5, p. 272–276, 2012.

SOH, L.-K.; TSATSOULIS, C. Texture analysis of sar sea ice imagery using gray level co-occurrence matrices. **IEEE Transactions on geoscience and remote sensing**, IEEE, v. 37, n. 2, p. 780–795, 1999.

SOLOMON, C.; BRECKON, T. **Fundamentals of Digital Image Processing**. [S.l.]: John Wiley & Sons, 2011. ISBN 9780470689783.

STOCK, D. World production and consumption of ceramic tiles. **Tile today**, Australian Tile Publications Pty Ltd., v. 73, p. 50–58, 2011.

TÉCNICAS, A. B. de N. **NBR 13818 - Placas cerâmicas para revestimento - Especificação e métodos de ensaios**. [S.l.], April 1997.

TRAINA, C. et al. Fast feature selection using fractal dimension. 2000.

TREVINO, A. **Introduction to K-means Clustering**. 2016. Disponível em: <https://www.datascience.com/blog/introduction-to-k-means-clustering-algorithm-learn-data-science-tutorials>.

UMANAND, L. **Nearest Neighbor Classifier and its variants**. 2012. Disponível em: <http://nptel.ac.in/courses/106108057/12>.

UPPULURI, A. **GLCM texture features**. 2008. Online. Acesso em: 14/04/17. Disponível em: <https://www.mathworks.com/matlabcentral/fileexchange/22187-glcm-texture-features>.

ZIOU, D.; TABBONE, S. et al. Edge detection techniques - an overview. **Pattern Recognition and Image Analysis C/C of Raspoznavaniye Obrazov I Analiz Izobrazhenii**, NAUKA/INTERPERIODICA PUBLISHING, v. 8, p. 537–559, 1998.