UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS CORNÉLIO PROCÓPIO
CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO

LUIZ GUSTAVO CASTILHO MACHADO

SIMULAÇÃO DE UM SENSOR DE FORÇA MULTIAXIAL UTILIZANDO MÉTODOS DE ANÁLISE DIGITAL DE IMAGENS E CLASSIFICADOR NEURAL

TRABALHO DE CONCLUSÃO DE CURSO

CORNÉLIO PROCÓPIO
2018

LUIZ GUSTAVO CASTILHO MACHADO

SIMULAÇÃO DE UM SENSOR DE FORÇA MULTIAXIAL UTILIZANDO MÉTODOS DE ANÁLISE DIGITAL DE IMAGENS E CLASSIFICADOR NEURAL

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina TCC 2, do curso de Engenharia de Controle e Automação da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Bacharel.

Orientador: Prof. Dr. Paulo Rogério Scalassara

CORNÉLIO PROCÓPIO

2018

Universidade Tecnológica Federal do Paraná
Campus Cornélio Procópio
Departamento Acadêmico de Elétrica
Curso de Engenharia de Controle e Automação

FOLHA DE APROVAÇÃO

Luiz Gustavo Castilho Machado

Simulação de um sensor de força multiaxial utilizando métodos de análise digital de imagens e classificador neural

Trabalho de conclusão de curso apresentado às 14:00hs do dia 29/11/2018 como requisito parcial para a obtenção do título de Engenheiro de Controle e Automação no programa de Graduação em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná. O candidato foi arguido pela Banca Avaliadora composta pelos professores abaixo assinados. Após deliberação, a Banca Avaliadora considerou o trabalho aprovado.

_____

Prof(a). Dr(a). Paulo Rogério Scalassara - Presidente (Orientador)

_____

Prof(a). Dr(a). Cristiano Marcos Agulhari - (Membro)

_____

Prof(a). Dr(a). Emerson Ravazzi Pires da Silva - (Membro)

A folha de aprovação assinada encontra-se na coordenação do curso.

DEDICATÓRIA

Dedico este trabalho à toda minha família que é a minha principal motivação para alcançar meus objetivos.

AGRADECIMENTOS

Em primeiro lugar, à Deus pelo dom da vida e por estar sempre presente.

À minha querida mãe, Angela Maria de Castilho, que nunca mediu esforços para criar e educar seus dois filhos.

À minha amada esposa, por me acompanhar, ser minha fortaleza durante todo este trajeto e me proporcionar o melhor presente da minha vida, meu filho Matheus.

À minha família, por todo o auxílio e carinho de sempre.

Ao professor Dr. Joel Perry, e à University of Idaho, por ter me proporcionado a oportunidade do desenvolvimento desse projeto e ter confiado a mim esta ideia.

Ao meu querido professor e orientador Dr. Paulo Rogério Scalassara, que paciente me guiou durante todo este projeto de uma forma muito profissional e extremamente paciente.

Aos meus amigos e irmãos de faculdade, Felipe Menegale, Yuri Andrade, Diego Alves, Abib Haddad e Veridiana Contieri, por todos os momentos vívidos dentro e fora da faculdade.

À Universidade Tecnológica Federal do Paraná, pela oportunidade de crescimento intelectual durante todos esses anos.

À CAPES, pela bolsa de estudo do Programa Jovens Talentos e do Programa Ciências Sem Fronteiras, que proporcionaram grandes experiências e fizeram imensa diferença em meu aprendizado profissional e pessoal.

E, finalmente, à todos que contribuíram de direta ou indiretamente para a realização deste trabalho.

RESUMO

MACHADO, Luiz G. C. **Simulação de um sensor de força multiaxial utilizando métodos de análise digital de imagens e classificador neural**. Trabalho de Conclusão de Curso (Graduação) – Engenharia de Controle e Automação. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2018.

Este trabalho desenvolve um conjunto de simulações acerca do funcionamento de um sensor de força multiaxial digital que realiza a medição das forças aplicadas à sua estrutura, por meio de técnicas de processamento e análise digital de imagens e classificação por um classificador neural (Rede Neural Artificial). O sensor proposto possui uma estrutura flexível que conecta duas superfícies concêntricas e dispostas paralelamente, onde uma das superfícies possui uma imagem pré-estabelecida e o outra possui um sensor de imagem. Toda a simulação é realizada virtualmente via software MATLAB, incluindo a implementação da análise digital do conjunto de amostras de imagens obtidas e a classificação do sentido e direção da força aplicada. Ao final, os resultados são analisados e são propostas melhorias para projetos futuros.

Palavras-chave: sensor de força multiaxial, análise digital de imagens, rede neural artificial.

ABSTRACT

MACHADO, Luiz G. C. **Simulation of a multiaxial force sensor using digital image analysis and neural classifier methods**. Final Project (Undergraduation) – Control and Automation Engineering. Universidade Tecnológica Federal do Paraná.Cornélio Procópio, 2018.

This work develops a set of simulations about the operation of a digital multiaxial force sensor that performs the measurement of the forces applied to its structure, through techniques of digital image processing and analysis and classification by a neural classifier (Artificial Neural Network). The proposed sensor has a flexible structure that connects two concentric and parallelly arranged surfaces where one of the surfaces has a predetermined image and the other has an image sensor. The entire simulation is performed virtually via MatLab software, including the implementation of the digital analysis of the set of image samples and the classification of the direction and direction of the applied force. At the end, the results are analyzed and improvements are proposed for future projects.

Keywords: multiaxial force sensor, digital image analysis, artificial neural network.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

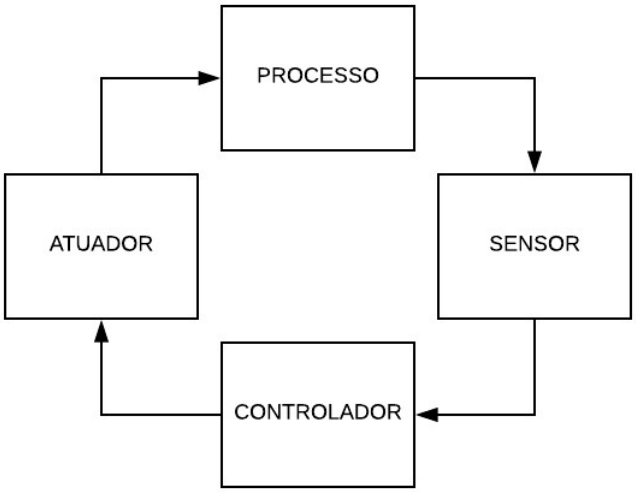
Após a Primeira Revolução Industrial, em meados do século XVIII, com o advento da máquina a vapor, a evolução da tecnologia empregada nos processos industriais vem crescendo exponencialmente. Os avanços surgem de modo a suprir as principais necessidades e interesses da indústria: a redução do custo e a maximização do lucro. A partir de tais necessidades, somadas às intenções de também reduzir o gasto empregado de bioenergia humana na realização das tarefas envolvidas nos processos industriais, surgiram os primeiros vestígios da automação industrial com a mecanização e o controle automático dos processos industriais (PERASSO, 2016; BEZERRA, 2018; SILVEIRA & LIMA, 2003).

Nos processos industriais, a grande necessidade se dá em controlar e manter constantes as variáveis envolvidas (variável de controle) – define-se como variáveis as condições a serem medidas (tais como carga, temperatura, pressão, vazão, nível, pH, posição, velocidade e etc) – para que se efetue o controle do processo (DUNN, 2013).

O controle de um processo pode ser realizado, basicamente, de dois modos, em malha aberta ou malha fechada. Nos sistemas de controle em malha aberta, o sinal de saída do processo não afeta a ação de controle – decisão do controlador. Como exemplo prático, tem-se uma máquina de lavar roupa, onde as operações de colocar de molho, lavar e enxaguar são executadas em sequência programada com relação ao tempo. Assim, a máquina de lavar roupa não quantifica quão limpas estão as roupas (sinal de saída) para mudar de um modo de operação para o outro. Nos sistemas de controle em malha fechada, o sinal de entrada (desejado) é comparado com o sinal de saída (real) e ação de controle é calculada com base nessa diferença – chamada de erro. Como exemplo prático, tem-se o controle da temperatura de um refrigerador. O sistema mede a temperatura do refrigerador, por meio de um termostato, compara com a temperatura desejada e efetua a ação de controle, ligando ou desligando o sistema de refrigeração (OGATA, 2000).

Uma das vantagens dos sistemas de controle em malha fechada é a robustez a perturbações externas ou variações dos parâmetros do sistema (OGATA, 2000). O diagrama da Figura 1 ilustra, genericamente, o funcionamento de um sistema de controle automático em malha fechada.

Figura 1 – Diagrama do funcionamento de um controle automático em malha fechada.



Fonte: Autoria própria.

Note que a principal diferença entre os sistemas de controle em malha fechada e os de malha aberta é a medição da variável de controle e sua comparação com o desejado. Assim, os sensores são elementos essenciais no controle de malha fechada.

De fato, nota-se a importância do sensor no cenário de controle automático de processos pela constante evolução da área de desenvolvimento de sensores. Pesquisadores tem desenvolvido sensores capazes de serem aplicados nas mais diversas áreas: sensores para detecção de aromas alimentícios, como abordado por (TIGGEMANN et al, 2014); sensores de ressonância magnética nuclear para análise do efeito da temperatura no relaxamento de sementes de macadâmia (CARVALHO; CABEÇA; CARÓSIO, 2018) e a análise da qualidade interna de frutas e outros produtos agropecuários (EMBRAPA, 2014); células de carga para medição de cargas atuantes em componentes (SILVA, 2012).

Com base neste contexto e na demanda no setor de desenvolvimento de sensores para o avanço no controle automático de processos, este trabalho propõe simular o desenvolvimento de um sensor que supra as necessidades dispostas na seção a seguir.

1.1 DEFINIÇÃO DO PROBLEMA

No mercado atual existem diversos tipos de sensores de força, os quais são capazes de medir a tensão mecânica aplicada em suas áreas sensíveis utilizando diferentes princípios de detecção. A massiva maioria baseia-se nos princípios da piezoresistividade, capacitância ou extensômetria. Os sensores mais comuns são capazes de realizar a detecção/medição da força apenas no eixo normal à sua superfície de detecção e, portanto, limitam-se à medição uniaxial. Nesse caso, para detecção em múltiplos eixos, seria necessária à associação de múltiplos sensores uniaxiais, demandando, também, mais espaço físico (HOFFMANN, 2017).

Alguns sensores existentes, conhecidos como células de carga, utilizam extensômetros para realizar a medição. Estes sensores são capazes de realizar a medição multiaxial, no entanto, apresentam custo relativamente elevado para uma faixa de detecção mais alta, com alguns modelos mais precisos podendo custar mais de R$5mil a unidade (BARBOSA et al, 2009; OMEGA, 2018; SCHMIDT, 2018).

Além disso, há a necessidade de se efetuar a compensação (calibração) para minimizar a interferência causada pelo efeito de Poisson – quando uma força é aplicada em uma única direção em um objeto, este pode ser esticado, tornando-se também mais fino, ou comprimido, tornando-se mais grosso – pois, ao aplicar uma força orientada a somente um dos eixos da célula de carga, haverá também uma força atuante nos outros eixos (HOFFMANN, 2017).

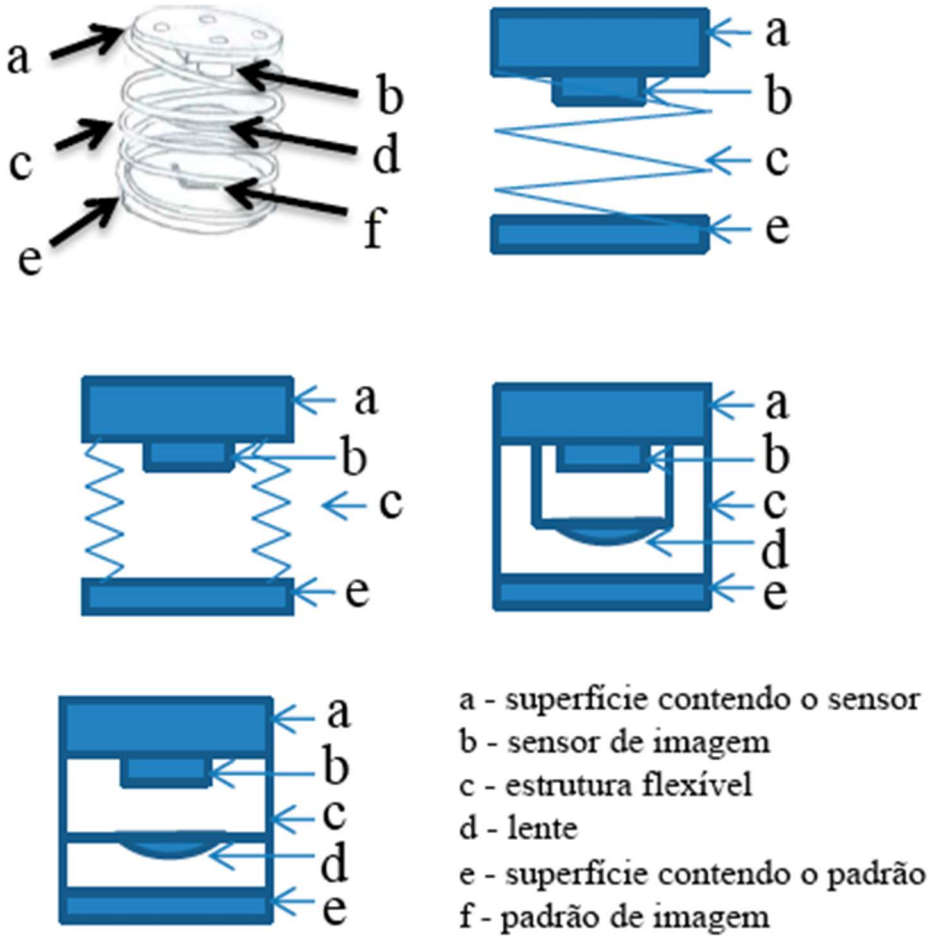
Diante deste cenário, vê-se uma necessidade no mercado de um sensor de força com a capacidade de detecção de forças em múltiplos eixos, com custo relativamente baixo e que possa atuar em uma ampla faixa de aplicação. Este trabalho desenvolve a simulação do funcionamento de um sensor de força multiaxial digital que realiza a medição das forças aplicadas à sua estrutura utilizando classificador neural e técnicas de processamento e análise de imagens digitais.

1.2 JUSTIFICATIVA

O sensor de força multiaxial digital baseia-se na detecção da deformação de uma imagem interna no sensor. A ideia, inovadora, consiste em utilizar uma estrutura composta por duas superfícies rígidas, conectadas por um corpo flexível (mola) e opostas paralelamente, onde

o movimento de uma superfície em relação a outra é resultante da aplicação de uma força à estrutura. Na parte interior da estrutura há um sensor de imagem, fixado em uma das superfícies rígidas, responsável por capturar um padrão de imagem fixado na superfície oposta, localizada na outra extremidade da estrutura. Ao submeter a estrutura a uma força, haverá uma mudança relativa de posição, entre o sensor de imagem e o padrão. A Figura 2 apresenta alguns esboços das ideias para o desenvolvimento da estrutura.

Figura 2 – Ideias de possíveis estruturas para o sensor.



Fonte: Autoria própria.

Por meio de um classificador neural e de técnicas de processamento de imagens, o sistema é capaz de calcular a deformação da imagem captada pelo sensor e detectar a direção e sentido da força aplicada na estrutura. Tendo o conhecimento das propriedades materiais da

estrutura flexível, o cálculo da intensidade da força resultante aplicada à estrutura também é possível.

Há um enorme mercado para a aplicação, como por exemplo:

- **Mercado Médico:** dispositivo de entrada intuitivo, para controlar equipamentos médicos (robôs de reabilitação, cadeira de rodas automáticas e etc), ferramenta para avaliação biomecânica e etc;
- **Mercado Industrial:** pesagem de produtos/equipamentos, medição da condição de aperto de garras em robôs industriais, ensaios para validação técnica e etc;
- **Informática:** dispositivos como controles, joysticks e etc.

1.3 OBJETIVOS

O principal objetivo desse trabalho é simular e validar o funcionamento de um sensor de força multiaxial, utilizando um sistema inteligente capaz de detectar a deslocamento de um padrão de imagem e apontar, por meio de um classificador neural, o sentido e orientação da força atuante em sua estrutura de detecção. Para isso, os seguintes objetivos específicos foram elaborados:

- Definir resolução do sensor de imagem;
- Definir as características do padrão de imagem
- Mapear todo o conjunto das possíveis imagens de detecção;
- Desenvolver estratégia para redução do tamanho do conjunto de dados;
- Definir configuração da Rede Neural Artificial para atuar como classificador;
- Simular o funcionamento;
- Validar a simulação.;
- Encontrar a melhor configuração da arquitetura da Rede Neural Artificial escolhida.

1.4 ESTRUTURA DO TRABALHO

A seguir é apresentada a fundamentação teórica dos conceitos abordados e necessários no decorrer deste trabalho, divididos em dois capítulos, os conceitos de reconhecimento de padrões em imagens e os de desenvolvimento de redes neurais artificiais. Após, são apresentados os materiais métodos utilizados, bem como a apresentação dos resultados obtidos com as simulações realizadas. Por fim, são feitas as considerações finais e apresentadas as ideias para continuidade e melhoria do projeto.

2 RECONHECIMENTO DE PADRÕES EM IMAGENS

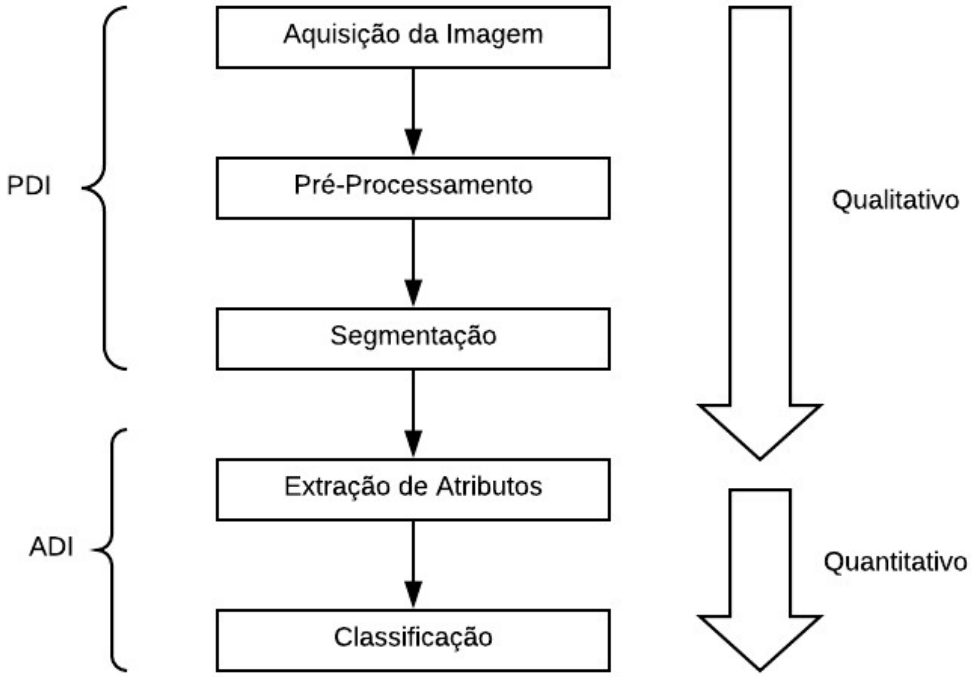
O problema de reconhecimento de padrões tem o objetivo de discriminar amostras de dados e classificar corretamente outras amostras apresentadas e, portanto, é um problema de classificação. A classificação é realizada com base em um conjunto de características (atributos) do objeto ou indivíduo, que são valorados. Existem, atualmente, diversas técnicas para o reconhecimento utilizando abordagens diferentes, como por exemplo a abordagem estatística, baseada em modelos probabilísticos e na teoria da decisão (aprendizagem bayesiana); a abordagem estrutural, baseada principalmente na teoria das linguagens formais (árvores de decisão); e a abordagem neural, que utiliza as propriedades das redes neurais para classificação (STANGE; NETO, 2010).

O processo de reconhecimento de padrões pode ser dividido em duas fases: a fase de aprendizagem, conhecida como treinamento, e a fase de classificação, também dita fase de reconhecimento. A fase de treinamento tem como objetivo determinar um conjunto de regras para definição dos padrões, utilizando um conjunto de exemplos (amostras) de treinamento. A fase de classificação tem como objetivo separar as amostras apresentadas em diferentes classes (grupos), conforme o conjunto de regras definido na fase de treinamento (STANGE; NETO, 2010; SILVA; SPATTI; FLAUZINO, 2010).

O fluxograma da Figura 3 apresenta o processo para o reconhecimento de padrões em imagens. Neste processo, há a associação de duas técnicas: a técnica de Processamento Digital de Imagens (PDI), onde ocorrem operações matemáticas para ajustes na imagem digital (processo qualitativo), e a técnica de Análise Digital de Imagens (ADI), onde ocorrem as medições em relação às regiões, partículas e objetos identificados na imagem digital (processo quantitativo). Deste modo, este processo padrão para o reconhecimento de padrões em imagens é conhecido como Processamento e Análise Digital de Imagens (PADI) (GOMES, 2001).

Nas subseções a seguir serão abordadas as etapas que compõem a técnica de Processamento e Análise Digital de Imagens.

Figura 3 – Etapas do Processamento e Análise Digital de Imagens



Fonte: Autoria própria.

2.1 AQUISIÇÃO DA IMAGEM

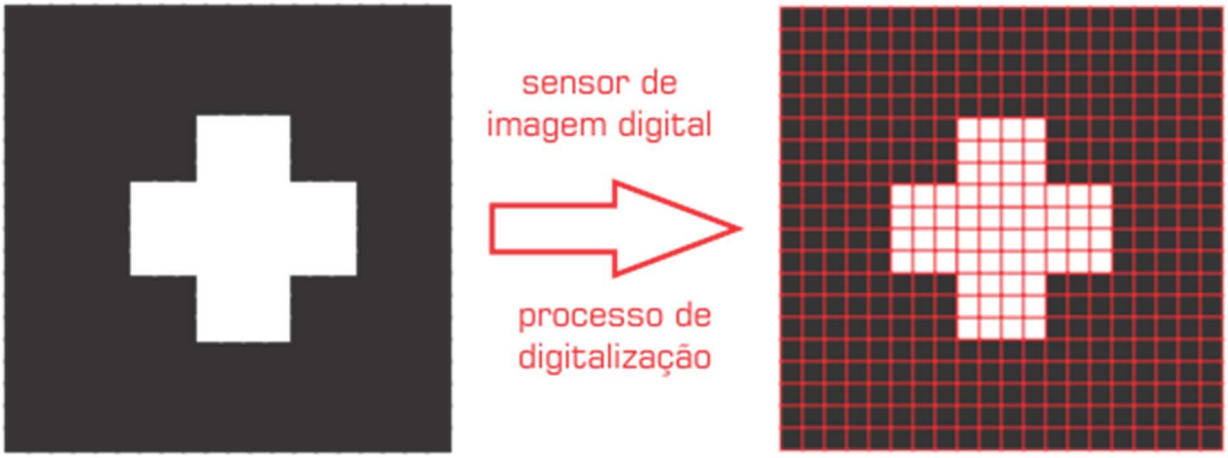
A primeira etapa do PADI é a aquisição da imagem, com a formação e digitalização da imagem. Uma imagem digital é composta por diversos *pixels* que representam a energia luminosa que incide sobre o objeto, onde cada *pixel* apresenta valor e localização específicos, relacionados à imagem real. Desta forma, imagens digitais podem ser representadas por vetores, ou matrizes, onde cada elemento representa a quantidade de energia do respectivo *pixel* da imagem (GONZALEZ; WOODS, 2010).

Matematicamente, uma imagem digital pode ser definida como uma função de f(x, y), onde x e y correspondem as coordenadas de cada um de seus *pixels*. Em imagens monocromáticas, isto é, em tons de cinza ou brilho, cada pixel recebe um valor proporcional a intensidade de energia luminosa naquele ponto. Em imagens coloridas, cada pixel recebe um conjunto de valores, conforme cada padrão de cor. Como exemplo, é possível citar o padrão RGB – que utiliza três canais de cores R (red – vermelho), G (green – verde) e B (blue – azul) – onde a combinação dos valores de seus canais resultam nas diversas cores conhecidas (GONZALEZ; WOODS, 2010).

A captura da energia luminosa é realizada por sensores de imagem capazes de transformar tal energia em energia elétrica. Os sensores *Charge Coupled Devices* (CCD) e *Complementary Metal-Oxide-Semiconductor* (CMOS) são os mais utilizados no mercado. Os sensores CCD utilizam capacitores para o sensoriamento, que atuam armazenando carga proporcionalmente à quantidade de luz recebida. Os sensores CMOS utilizam do mesmo princípio – possuem elementos para armazenar carga proporcionalmente à quantidade de luz recebida – no entanto, os elementos responsáveis por essa tarefa são os fotodiodos, que formam os pixels da imagem digital (BAPTISTA, 2016).

A Figura 4 ilustra o processo de aquisição de uma imagem digital:

Figura 4 – Ilustração do processo de aquisição de uma imagem digital.



Fonte: Autoria própria.

Os resultados obtidos pelo PADI são influenciados diretamente pelos resultados de cada uma de suas etapas. Deste modo, para a obtenção de bons resultados, é imprescindível o cuidado no desenvolvimento de suas etapas. Portanto, a aquisição de imagens com boa qualidade minimiza o trabalho demandado nas próximas etapas, do mesmo modo que a etapa seguinte, o pré-processamento, se realizado de maneira eficiente, reduzirá o esforço demandado nas etapas conseguintes (GONZALEZ; WOODS, 2010; AUGUSTO, 2012; GOMES, 2001).

Neste trabalho são assumidas imagens ideais para a simulação dos resultados da classificação. Isto é, não são considerados problemas com ruídos, iluminação, foco e outros problemas comumente originados na aquisição da imagem.

2.2 PRÉ-PROCESSAMENTO

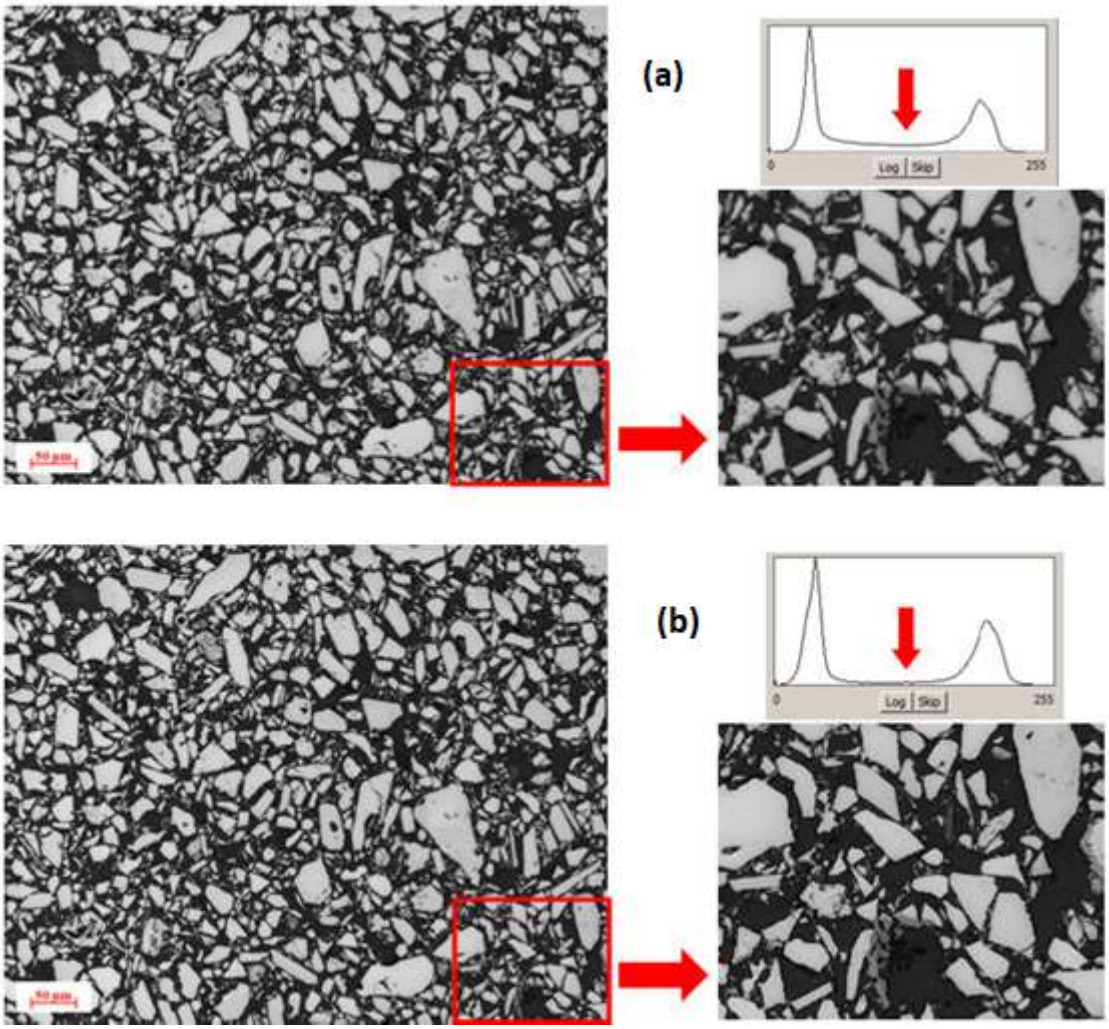
A etapa de pré-processamento tem por finalidade corrigir falhas provenientes da etapa anterior (aquisição da imagem) ou mesmo realçar detalhes importantes na análise em determinadas aplicações. Podem haver inúmeros defeitos originados na etapa anterior, como também existem inúmeras possibilidades de corrigí-los. As técnicas utilizadas nesta etapa apresentam soluções específicas, assim técnicas que apresentam resultados satisfatórios para um tipo de problema podem não ter a mesma eficácia para outros tipos de problemas (GONZALEZ; WOODS, 2010).

Nesta etapa é comum a utilização de métodos matemáticos para melhora no brilho e contraste da imagem, redução de ruídos, correção de iluminação, realce de bordas, correção de fundo, entre outros (GOMES, 2001).

Um exemplo de pré-processamento amplamente utilizado é o delineamento, que tem como objetivo o realce de bordas na imagem para facilitar a etapa seguinte, a segmentação. Gonzalez e Woods (2010) definem borda como sendo a divisão entre duas regiões com características relativamente distintas de nível de cinza, de modo que uma borda é melhor definida quanto maior for essa diferença entre as intensidades dessas regiões. No entanto, é comum o acontecimento do “efeito halo”, devido à limitação da resolução do sistema de aquisição das imagens, onde as bordas se tornam mais suaves, tendo valores intermediários às regiões de suas fronteiras.

Para correção do “efeito halo”, o processo de delineamento faz uma varredura na imagem, detecta as bordas por meio da comparação dos níveis de cinza de pixels vizinhos e decide a qual região estes pixels pertencem, baseando-se na maior proximidade de valores entre o pixel em questão e as regiões em questão e alterando o valor deste pixel para o mesmo valor da região ao qual ele foi incluso (GONZALEZ; WOODS, 2010). A Figura 5 apresenta a diferença entre a imagem original e o resultado após a aplicação do delineamento:

Figura 5 – Aplicação do filtro de delineamento: (a) imagem original; (b) imagem após aplicação do filtro.



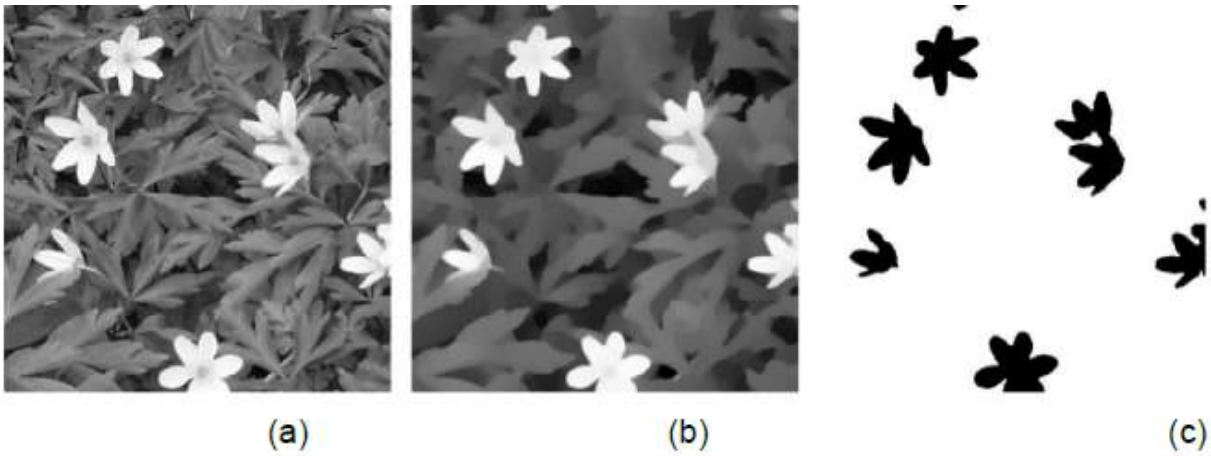
Fonte: (AUGUSTO, 2012).

2.3 SEGMENTAÇÃO

Na etapa de segmentação, o objetivo principal é subdividir, em regiões ou objetos de interesse, a imagem em questão com o intuito de obter informações para extração. A detecção de tais regiões ou objetos na imagem é definida com auxílio de algumas características como cor, borda ou similaridade (SOLEM, 2012, apud RIBEIRO, 2015).

Nos casos em que a segmentação é baseada pela característica da cor, a eficiência está relacionada com o fato da cor do objeto de interesse ser consideravelmente diferente da cor das outras regiões. Nestes casos é possível a utilização do limiar de cor, onde são mantidos os pixels que possuem seus valores enquadrados dentro da faixa do limiar desejado (RIBEIRO, 2015). A Figura 6 ilustra o processo de segmentação:

Figura 6 – Processo de segmentação: (a) imagem original; (b) imagem após aplicação do filtro da média; (c) imagem após o processo de segmentação pelo limiar de cor.



Fonte: (SOLEM, 2012, apud RIBEIRO, 2015).

Note que na imagem original (Figura 6-a), as flores destacam-se das demais regiões e objetos, pois seus tons são consideravelmente diferentes. No entanto, é possível notar que algumas pequenas regiões apresentam tons semelhantes, o que resultaria na imagem final com regiões indesejadas. Então, para corrigir o problema, foi aplicado um filtro da média, que faz a varredura dos pixels da imagem, calculando e atribuindo o valor de cada pixel como a média do valor de seus pixels vizinhos. Desta forma, os tons da imagem resultante se tornam mais homogêneos (note que há a perda de alguns detalhes) e as pequenas regiões em tons de branco, se tornam mais escuras, assim aumentando a eficácia da segmentação utilizando o limiar de cor.

2.4 EXTRAÇÃO DE ATRIBUTOS (CARACTERÍSTICAS)

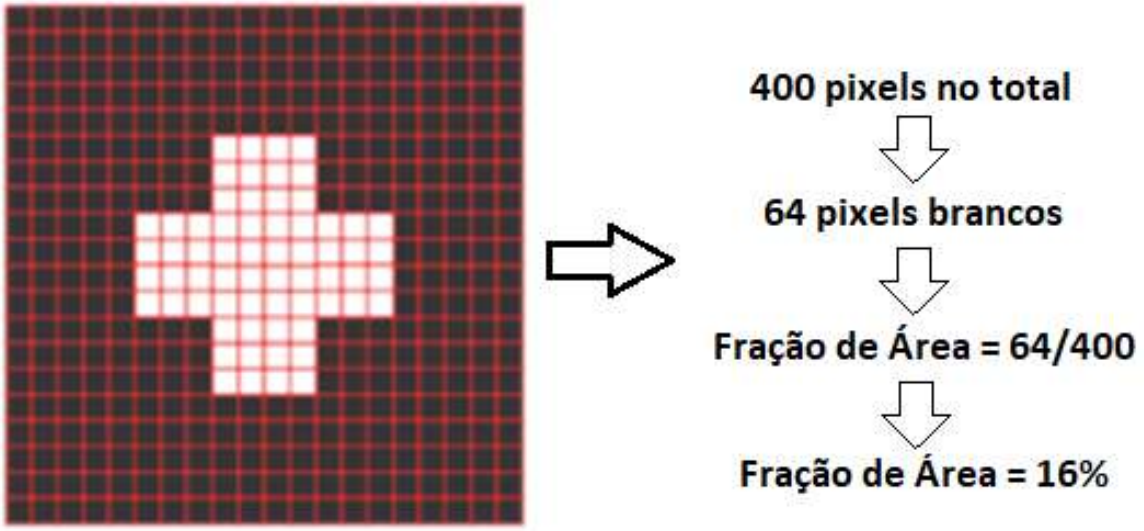
A análise digital de imagens é iniciada na etapa de extração de características, onde são coletadas métricas da imagem resultante das etapas anteriores. Por meio dessas métricas os pixels são descritos por atributos característicos e, assim, são gerados dados quantitativos para serem utilizados na aplicação final (RIBEIRO, 2015; AUGUSTO, 2012; GONZALEZ; WOODS, 2010).

Segundo Gomes (2001) existem dois tipos de medidas a serem extraídas: os parâmetros de campo e os parâmetros de região. Os parâmetros de campo se referem a imagem como um todo, tais como número de objetos, área e fração de área ocupada pelos objetos. Os

parâmetros de região se referem aos objetos individualmente, extraindo-se parâmetros de cada objeto, tais como tamanho, forma e posição do respectivo objeto (AUGUSTO, 2012).

Em particular, neste trabalho será utilizada uma métrica de região para a extração das características desejadas. O tipo de métrica que será utilizada é a área ocupada, que realiza a contagem dos pixels brancos, isto é, faz a soma dos valores dos pixels da imagem binária. Uma outra medida derivada é a fração de área, que é obtida pela razão entre o número de pixels brancos e o número total de pixels. A Figura 7 mostra o processo de obtenção desta métrica:

Figura 7 – Obtenção da métrica de fração de área.



Fonte: Autoria própria.

2.5 CLASSIFICAÇÃO

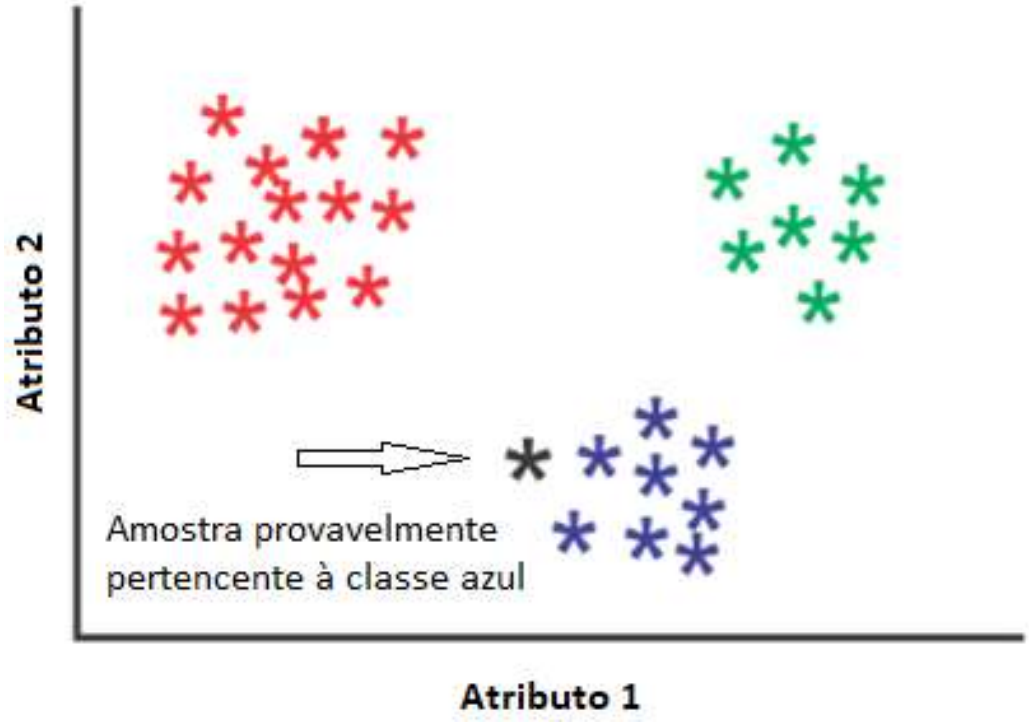
A etapa de reconhecimento de padrões é a etapa final do PADI. Um padrão é definido como uma descrição quantitativa em relação à uma região de interesse em uma imagem e uma classe de padrões é um agrupamento de padrões que apresentam características em comum. Assim, o reconhecimento de padrões é a tarefa de atribuir padrões às suas respectivas classes de forma automática (GONZALEZ; WOODS, 2010).

Deste modo, essa técnica é utilizada para realizar a classificação a partir do conjunto de atributos obtidos, construindo um espaço de atributos onde cada objeto é dado por um vetor contendo os atributos que o caracterizam. A classificação é realizada analisando-se o

posicionamento desses vetores no espaço de atributos, onde os grupos formados por pontos próximos, ou seja, características similares, indicam as classes de divisão (GOMES, 2001).

O reconhecimento de padrões pode ser dividido em dois grupos: a classificação supervisionada e a classificação não-supervisionada. Na classificação supervisionada, são formados grupos de padrões conhecidos e as classes são determinadas de acordo com sua proximidade a esses grupos. Na classificação não-supervisionada, não há grupos de padrões conhecidos e, portanto, a classificação é dada por meio dos agrupamentos dos pontos próximos. (GOMES, 2001). A Figura 8 ilustra como ocorre a classificação supervisionada, que será o tipo de classificação utilizada nesse trabalho:

Figura 8 – Ilustração do processo de classificação supervisionada.



Fonte: Autoria própria.

3 REDES NEURAIS ARTIFICIAIS (RNA)

As Redes Neurais Artificiais (RNA) “são modelos computacionais inspirados no sistema nervoso de seres vivos. Possuem a capacidade de aquisição e manutenção do conhecimento” (SILVA; SPATTI; FLAUZINO, 2010). Tais modelos são baseados na estrutura do cérebro humano e na maneira em que ele processa informações. Haykin (2001) afirma que o cérebro humano é um sistema de processamento de informação altamente complexo e não-linear, que processa informações paralelamente.

O processamento de informações no cérebro humano é realizado em suas bilhões de células estruturais – mais conhecidas como neurônios – que, quando interligadas entre si, constituem o sistema nervoso. De maneira análoga, as RNA são formadas por um conjunto de unidades de processamento, interligadas entre si, conhecidas como neurônios artificiais (HAYKIN, 2001).

As principais características de uma RNA são: a adaptação por experiência, com o ajuste dos parâmetros a cada iteração; a capacidade de aprendizado por meio de treinamentos pelos quais a rede é capaz de detectar a relação entre as diversas variáveis que compõem o sistema; a habilidade de generalização, que a torna capaz de generalizar o conhecimento, adquirido por meio do treinamento, e estimar soluções mesmo que desconhecidas, isto é, soluções que não estão contidas no grupo de soluções de treinamento; a organização de dados por meio de agrupamento de padrões que apresentam particularidades em comum; o armazenamento distribuído do conhecimento entre as diversas sinapses existentes, garantindo robustez caso haja falhas em alguns neurônios; e a relativa facilidade de prototipagem, pelo fato de os cálculos utilizarem, normalmente, operações elementares para encontrarem os resultados (SILVA; SPATTI; FLAUZINO, 2010).

A gama de aplicações de RNA é diversa, podendo citar-se aplicações como o aproximador universal de funções, controle de processos, clusterização (agrupamento de dados), sistemas de previsão, otimização de sistemas, memórias associativas e, como no caso deste trabalho, o reconhecimento e classificação de padrões (SILVA; SPATTI; FLAUZINO, 2010; BERG & MULLER, 1993; BUENO, 2016; CASTRO, 1995; ERPEN, 2004; NISHIDA, 1998; SILVA, 2005).

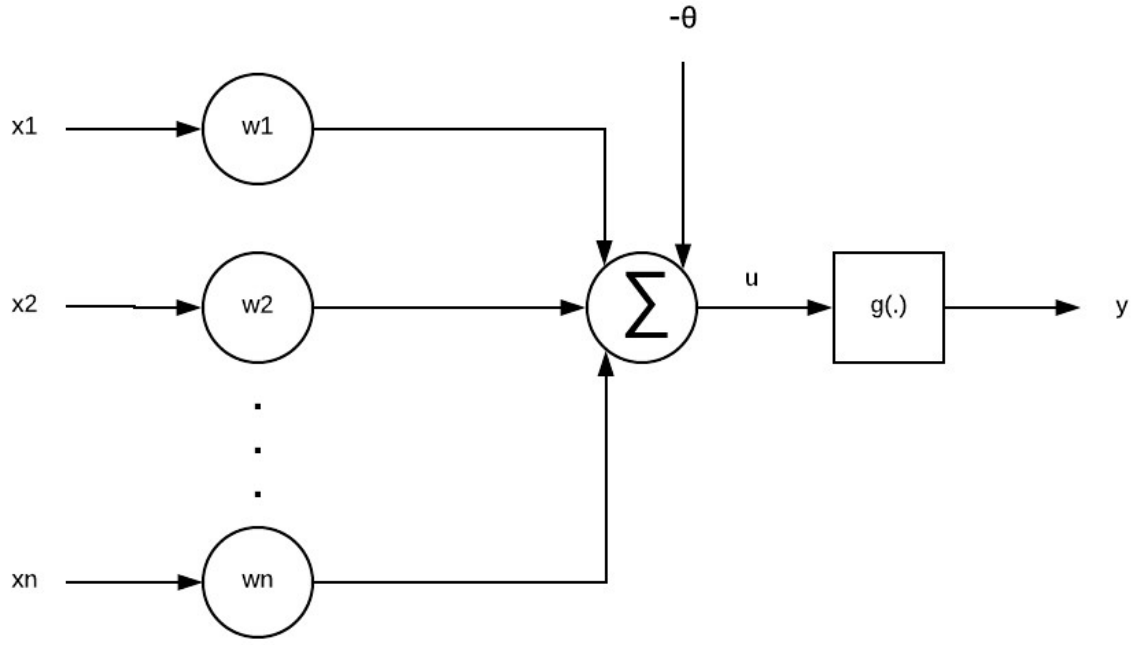
A seguir será apresentado o embasamento teórico sobre o funcionamento de um neurônio artificial, de algumas arquiteturas de RNA e alguns processos de aprendizado, e da topologia de RNA utilizada neste trabalho, denominada Perceptron Multicamadas.

3.1 NEURÔNIO ARTIFICIAL

Os neurônios artificiais realizam funções simples de coleta de sinais de entrada e produção de resposta de acordo com suas especificações operacionais. São modelos não-lineares e, geralmente, têm saídas contínuas (SILVA; SPATTI; FLAUZINO, 2010).

De acordo com Silva, Spatti e Flauzino (2010), o modelo de neurônio mais utilizado nas diferentes arquiteturas de RNA, foi proposto por McCulloch & Pits, em 1943, englobando as principais características de uma rede neural biológica: paralelismo e alta conectividade. A Figura 9 apresenta o modelo proposto por McCulloch & Pits.

Figura 9 – Representação de um neurônio artificial



Fonte: Autoria própria.

Nesse modelo, as entradas são representadas pelo conjunto $\{x_1, x_2, ..., x_n\}$ e são ponderadas pelo conjunto $\{w_1, w_2, ..., w_n\}$, denominado conjunto dos pesos sinápticos. Deste modo, é dada a relevância relativa a cada entrada. O resultado da combinação linear ($\sum$) das entradas x_i e pesos sinápticos w_i é então comparado ao limiar de ativação (θ), para que se possa

gerar um valor de disparo u. Assim, se u for positivo, então o neurônio artificial produz um potencial excitatório, caso contrário, o potencial será inibitório. Por fim, o sinal de saída é limitado pela função de ativação (g) para que a saída seja condicionada dentro de um intervalo de valores desejado.

3.2 ARQUITETURA DE RNA

É definida como arquitetura de uma RNA a forma como os neurônios são dispostos e conectados entre si, por meio das conexões sinápticas. Para a análise das diferentes arquiteturas abordadas neste trabalho, é importante entender a divisão das RNA em três diferentes camadas, denominadas camada de entrada, camada intermediária e camada de saída (HAYKIN, 2001).

A camada de entrada é a camada na qual há a recepção das informações, geralmente normalizadas em relação às faixas de variação dinâmica das funções de ativação, assim possibilitando maior precisão numérica para as operações numéricas da rede. A camada intermediária tem a função de extrair as características associadas ao processo, sendo responsável pela maior parte do processamento interno da rede. A camada de saída é responsável pela produção e apresentação dos resultados da rede, que foram processados pelas camadas anteriores (SILVA; SPATTI; FLAUZINO, 2010).

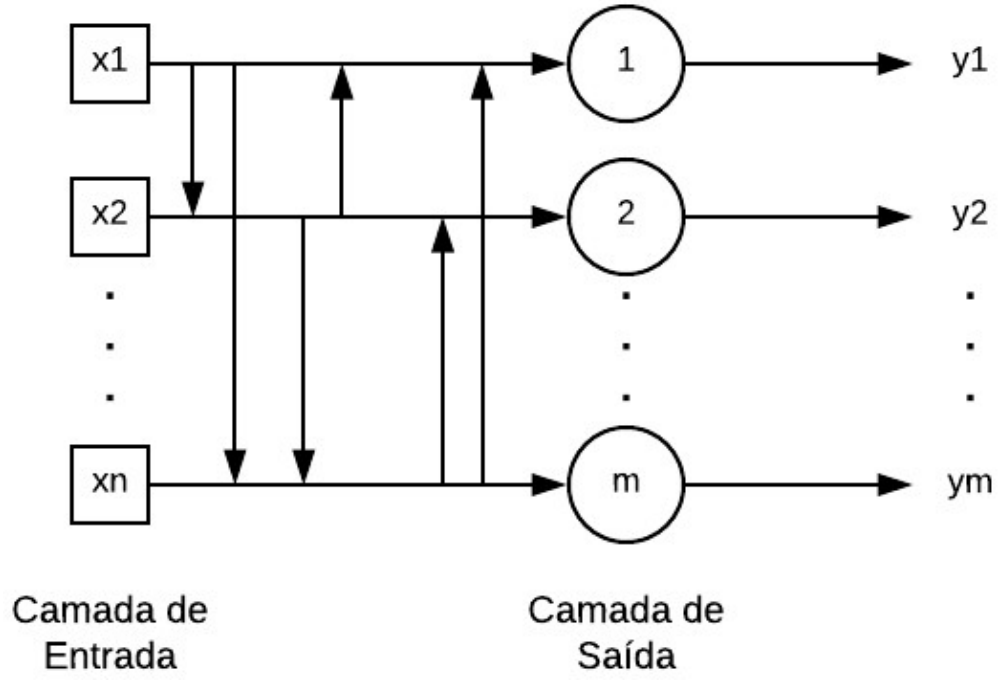
Analisando as arquiteturas em relação à disposição das camadas de neurônios e suas conexões é possível fazer a divisão em quatro principais arquiteturas: *feedforward* de camada simples, *feedforward* de camadas múltiplas, recorrente (com realimentação) e reticulada.

3.2.1 *FEEDFORWARD* DE CAMADA SIMPLES

A arquitetura *feedforward* de camada simples, ilustrada na Figura 10, caracteriza-se por uma camada de entrada e apenas uma outra camada de neurônios, sendo a própria camada de saída – as entradas possuem conexões com todos os neurônios da camada de saída. Assim, o fluxo de informações parte da camada de entrada em direção à camada de saída, de forma unidirecional. Portanto, a quantidade de neurônios será sempre igual ao número de saídas (SILVA; SPATTI; FLAUZINO, 2010).

Esse tipo de arquitetura é empregado tipicamente em problemas de classificação de padrões e filtragem linear. As principais topologias de RNA que utilizam essa arquitetura são a Adaline e a Perceptron. Tais topologias são largamente difundidas e não serão abordadas nesta fundamentação. Para maiores informações sobre elas, o autor deste trabalho recomenda a leitura de Silva, Spatti e Flauzino (2010) e Haykin (2001).

Figura 10 – Representação de uma rede *feedforward* de camada simples.



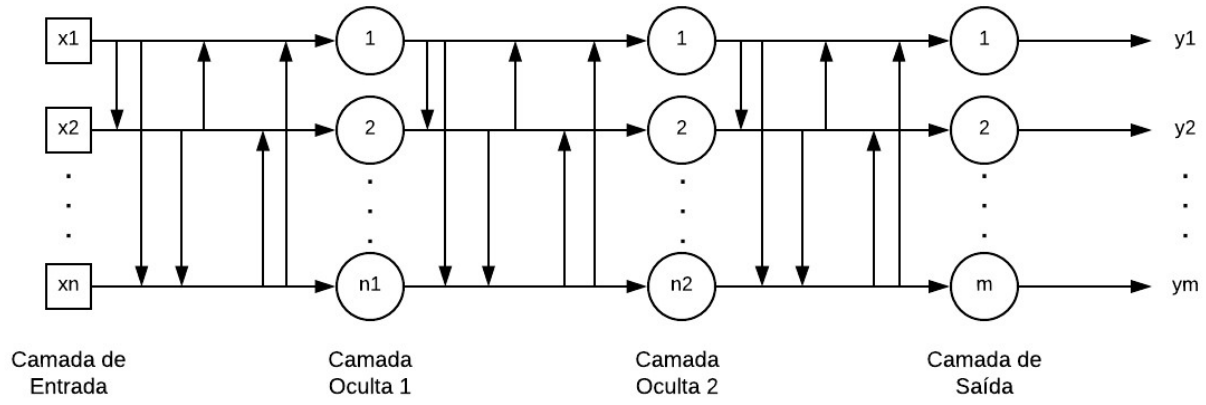
Fonte: Autoria própria.

3.2.2 *FEEDFORWARD* CAMADAS MÚLTIPLAS

A arquitetura *feedforward* de camadas múltiplas, ilustrada na Figura 11, caracteriza-se por uma camada de entrada, pelo menos uma camada intermediária, dita escondida ou oculta, e uma camada de saída. O número de camadas escondidas depende do tipo, da complexidade e, também, da quantidade e qualidade dos dados disponíveis para o problema. Deste modo, a quantidade de neurônios que compõe a primeira camada escondida não é necessariamente igual ao número de entradas da RNA. No entanto, a quantidade de saídas é sempre igual a quantidade de neurônios contidos na camada de saída (SILVA; SPATTI; FLAUZINO, 2010).

A gama de aplicação deste tipo de arquitetura relaciona problemas diversos. Como exemplo, tem-se a aproximação de funções, identificações de sistemas, otimização, robótica, controle de processos e classificação de padrões. As principais topologias de RNA que utilizam essa arquitetura são a Perceptron Multicamadas e a Rede de Base Radial (RBF). Neste trabalho, será utilizado a topologia Perceptron Multicamadas (PMC) como um classificador neural. A topologia PMC será fundamentada mais adiante neste trabalho, na Seção 3.4.

Figura 11 – Representação de uma rede *feedforward* de camadas múltiplas.



Fonte: Autoria própria.

3.3 PROCESSO DE APRENDIZADO

Uma das principais características das RNA é a capacidade de aprender a partir de treinamentos, os quais possibilitam detectar relações entre as diversas variáveis que compõe o problema. Os treinamentos são feitos a partir da apresentação de amostras de dados que exprimem o comportamento do sistema. Caso o treinamento seja bem-sucedido, a rede será capaz de generalizar soluções de modo a produzir saídas próximas das desejadas, a partir de qualquer entrada fornecida à rede (SILVA; SPATTI; FLAUZINO, 2010).

Quando uma RNA é estimulada por um ambiente, sofre alterações em seus parâmetros como resultado dessa estimulação. Assim, a RNA responde de uma maneira nova ao ambiente devido as alterações internas sofridas em sua estrutura (HAYKIN, 2001).

A partir de um algoritmo de aprendizagem, que consiste em passos para o ajuste dos pesos sinápticos e limiares, a RNA será “capaz de extrair características discriminantes do

sistema a ser mapeado por intermédio de amostras que foram retiradas do seu contexto" (SILVA; SPATTI; FLAUZINO, 2010).

A seguir, são apresentadas duas técnicas de treinamento de RNA: o Treinamento Supervisionado e o Treinamento Não-Supervisionado.

3.3.1 TREINAMENTO SUPERVISIONADO

No Treinamento Supervisionado, o conjunto de amostras de dados apresentados a RNA para o treinamento é acompanhado da saída desejada para cada amostra. Assim, as amostras de dados apresentadas não precisam necessariamente caracterizar o comportamento do sistema, devido à capacidade de generalização de uma RNA (SILVA; SPATTI; FLAUZINO, 2010).

Com isso, a cada amostra apresentada à RNA, os pesos sinápticos e limiares são supervisionados, por meio de comparações, e são atualizados utilizando algoritmos de aprendizagem que buscam corrigir a discrepância entre as saídas produzidas pela rede e as saídas desejadas.

3.3.2 TREINAMENTO NÃO-SUPERVISIONADO

No Treinamento Não-Supervisionado, o conjunto de amostras apresentadas à RNA para o treinamento não possui a saída desejada para cada amostra. Assim, a RNA adapta-se de acordo com as características comuns existentes entre as amostras, de modo a formar diferentes grupos com elementos que possuem similaridades. Após a divisão em grupos de similaridade, os pesos sinápticos e limiares são ajustados pelo algoritmo de aprendizagem. A quantidade máxima desses possíveis grupos, conhecidos como *clusters*, pode ser especificada pelo projetista (SILVA; SPATTI; FLAUZINO, 2010).

3.4 PERCEPTRON MULTICAMADAS

A topologia Perceptron Multicamadas (PMC), como o próprio nome sugere, utiliza os conceitos da arquitetura *feedforward* de camadas múltiplas. Assim, uma PMC é constituída

de pelo menos uma camada intermediária de neurônios e, diferente de topologias mais simples, sua camada de saída pode ser composta por mais de um neurônio, tendo, assim, diversas saídas possíveis (HAYKIN, 2001).

As redes com camadas múltiplas possuem melhor robustez em relação as redes com camada simples pelo fato de serem capazes de extrair características de ordem elevada. Os neurônios ocultos, isto é, a camada oculta, são responsáveis por extrair a não-linearidade dos dados (SILVA; SPATTI; FLAUZINO, 2010).

A classificação de padrões é uma das principais aplicações de uma PMC. Ela é utilizada para a classificação de padrões não linearmente separáveis, isto é, ela é capaz de separar padrões no hiperespaço que não podem ser separados por uma única linha reta. Sua grande versatilidade e robustez se dá, principalmente, pelo desenvolvimento do processo de aprendizado baseado no algoritmo de retropropagação do erro, apresentado a seguir.

3.4.1 ALGORITMO DE RETROPROPAGAÇÃO (BACKPROPAGATION)

O processo de aprendizado da RNA Multicamadas é dado por meio do algoritmo de Retropropagação de Erro, baseado na regra de correção de erro utilizando o método do gradiente descendente – as correções são proporcionais ao gradiente do erro – sendo o algoritmo de treinamento supervisionado mais difundido e um dos principais responsáveis pela eficácia e robustez da topologia multicamadas (HAYKIN, 2001).

Basicamente, o algoritmo de retropropagação calcula o erro da resposta real em relação à resposta desejada, à medida que as amostras de dados vão sendo apresentadas a rede, e retropropaga esse erro ajustando os pesos das conexões sinápticas de forma a aproximar a resposta da rede à resposta desejada e, portanto, a minimizar esse erro. Assim, o algoritmo de retropropagação é definido com base em um método de minimização (otimização) de uma função custo (HAYKIN, 2001).

Devido ao Algoritmo Backpropagation ser amplamente difundido e abordado por diversos autores, o mesmo não será abordado mais profundamente neste trabalho. Além disso, o objetivo aqui é simular o funcionamento do sensor proposto e, para isso, tem auxílio do software MATLAB e suas toolbox que implementam automaticamente este algoritmo. Para maiores detalhes sobre o Algoritmo Backpropagation, consulte Haykin (2001) e Silva, Spatti e Flauzino (2010).

4 MATERIAIS E MÉTODOS

Para simular o funcionamento do sensor multiaxial, de baixo custo, baseado em um sensor de imagem e um sistema inteligente capaz de processar a imagem e classificar a direção e sentido da força ao qual está submetido, foram utilizados os seguintes materiais, durante todo o desenvolvimento:

- Notebook
  - Processador: Intel Core i7-4500U, de 2.40GHz
  - Memória RAM: 8GB
- Software MATLAB.

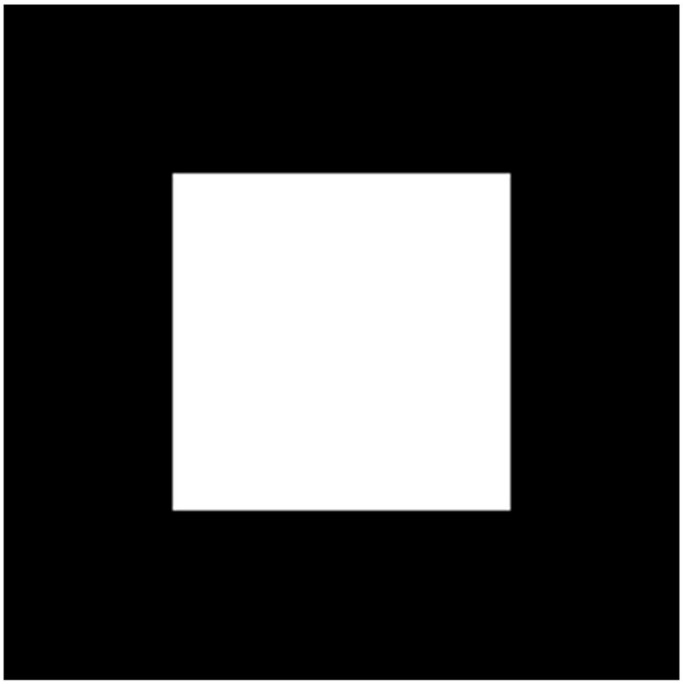
O sensor idealizado possui uma estrutura composta por duas superfícies rígidas e um corpo flexível, de modo que as duas superfícies são dispostas opostamente, de forma paralela. Uma das superfícies contém uma imagem e a outra o sensor de imagem. Todo o conceito do funcionamento do sensor baseia-se na ideia de coletar imagens dentro da estrutura flexível do sensor, para mensurar o deslocamento – relativo entre as superfícies – da imagem ao aplicar-se uma força à estrutura.

Sabendo que o sensor de imagem é o componente que apresenta o maior custo relativo no projeto, a escolha de um modelo de sensor mais barato reduz consideravelmente seu custo. Assim, durante as simulações foi considerado um sensor de baixa resolução, com resolução de aproximadamente 0.3MPixels (640 x 480 pixels) similar ao módulo câmera VGA OV7670 encontrado no mercado por um preço acessível.

Com a definição da resolução do sensor de imagem, o passo seguinte consistiu na definição do padrão de imagem a ser detectado. A Figura 12 mostra o formato escolhido empiricamente.

A sensibilidade do sensor é diretamente relacionada com o coeficiente de elasticidade da estrutura. Assim, para uma mesma força aplicada, quanto maior o coeficiente de elasticidade da estrutura, menor será o deslocamento da imagem relativo à posição inicial pois maior será a força contrária ao movimento, conforme comprovado pela Lei de Hooke. A área de detecção é delimitada pela sensibilidade máxima do sensor. Assim, determinou-se, empiricamente, a área de detecção quadrada medindo 200 por 200 pixels e o padrão de imagem, mostrado na Figura 12, com 100 por 100 pixels.

Figura 12 – O padrão de imagem.



Fonte: Autoria própria.

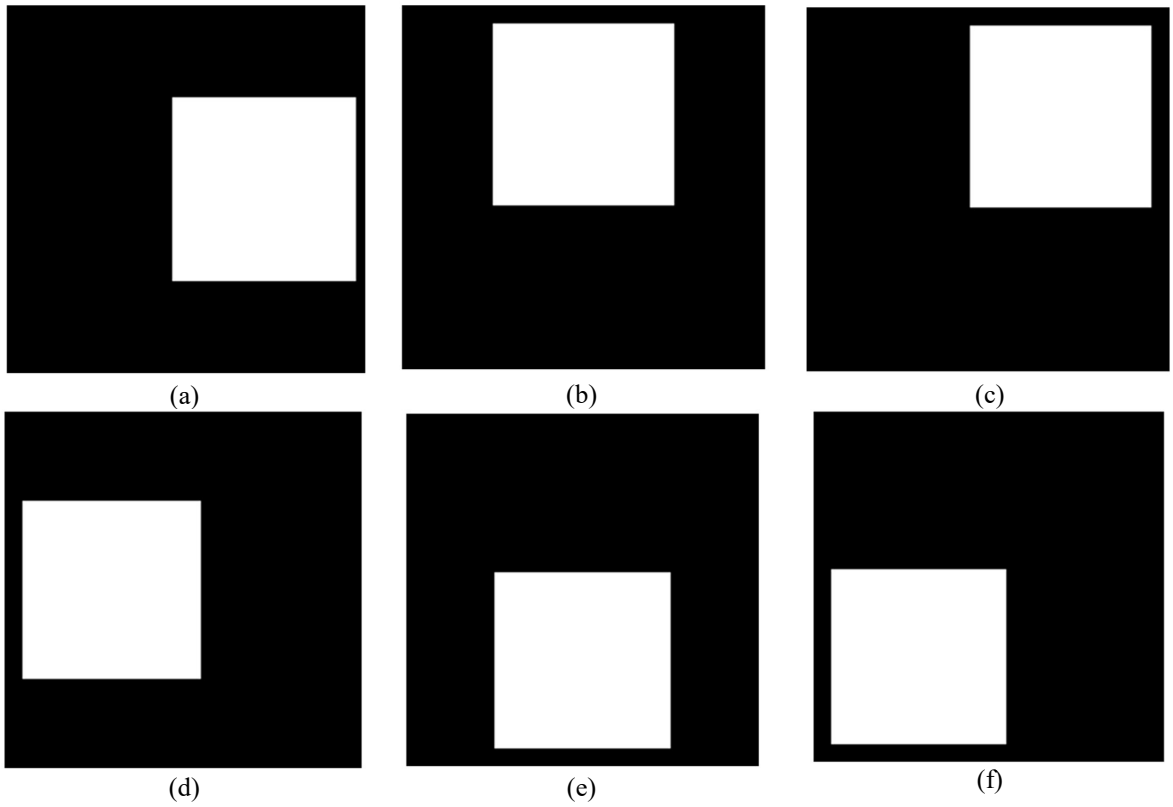
Sabe-se que o processamento e a análise digital de imagens têm início na aquisição da imagem – vide fluxograma da Figura 3, abordado no Capítulo 2 deste trabalho. Como o objetivo desse trabalho é simular e validar o funcionamento, a aquisição das imagens foi feita no ambiente do software MATLAB e foram desprezadas as etapas que constituem o processamento digital das imagens (pré-processamento e segmentação), assumindo, assim, condições ideais. Deste modo, os pixels das imagens geradas pelo software apresentam apenas dois valores possíveis: 0 ou 1 (preto ou branco).

Desta forma, todas as possíveis imagens resultantes do deslocamento do padrão de imagem foram mapeadas, com o objetivo de, posteriormente, serem utilizadas no treinamento e validação do classificador neural. Para isso, criou-se uma função que reproduz o padrão e é capaz de deslocá-lo, pixel a pixel. A função tem como entrada os parâmetros de deslocamentos nos eixos x, y e z. O deslocamento é relativo ao centro do padrão, posicionado exatamente no centro da imagem.

Os eixos foram dispostos de tal forma que o eixo x é disposto horizontalmente, com deslocamento positivo para a direita, e o eixo y verticalmente, com deslocamento positivo para cima – tomando como referência o plano desta página e a posição do leitor. O eixo z é disposto

perpendicularmente a esta página, de modo que um deslocamento positivo em z significa um padrão de imagem maior (mais próximo do leitor) e, por sua vez, um deslocamento negativo em z é traduzido como um padrão de imagem menor (mais distante do leitor). A Figura 13 ilustra os deslocamentos relativos aos eixos x e y:

Figura 13 – Deslocamento nos sentidos (a) X+, (b) Y+, (c) X+Y+, (d) X–, (e) Y–, (f) X–Y–.



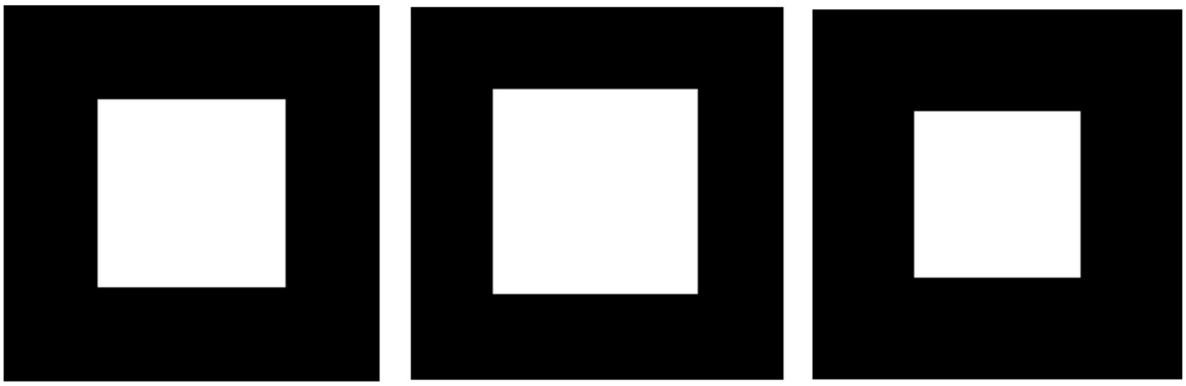
Fonte: Autoria própria.

Tendo a área de detecção e o tamanho do padrão de imagem estabelecidos, torna-se possível obter a faixa de deslocamento. Considerando a área de detecção quadrada, com cada lado desta área contendo 200 pixels, e o padrão de imagem quadrado, com cada um de seus lados contendo 100 pixels, posicionados de forma concêntrica, restam ainda 100 pixels para o deslocamento bidirecional (eixos x e y). Ao estabelecer que o deslocamento pode ser positivo ou negativo, têm-se o módulo máximo de deslocamento igual a 50 pixels.

Para o deslocamento em z, considerou-se que a cada unidade de deslocamento em z, 02 (duas) camadas (fileiras) de pixels seriam acrescentadas (para o deslocamento positivo, isto é, aproximação do padrão ao sensor de imagem), ou removidas (para o deslocamento

negativo, isto é, afastamento do padrão ao sensor de imagem), de cada lateral do padrão, mantendo o formato inicial e sua simetria. Além disso, considerou-se o módulo de deslocamento no eixo z sendo em no máximo 5 unidades. A Figura 14 ilustra os deslocamentos relativos ao eixo z:

Figura 14 – (a) Mostra o padrão de imagem (b) Deslocamento Z+ e (c) Deslocamento Z–.



(a) (b) (c)

Fonte: Autoria própria.

Assim, tendo definido a orientação dos eixos e seus deslocamentos, iniciou-se o processo de varredura de todas as possíveis imagens.

Com todo o conjunto de dados do problema definido, prosseguiu-se para a etapa da classificação. Para atuar na classificação da direção e sentido da força aplicado a estrutura, a topologia de RNA escolhida foi a Perceptron Multicamadas (PMC) por se tratar de uma topologia amplamente difundida e apresentar a possibilidade de utilização de múltiplas saídas.

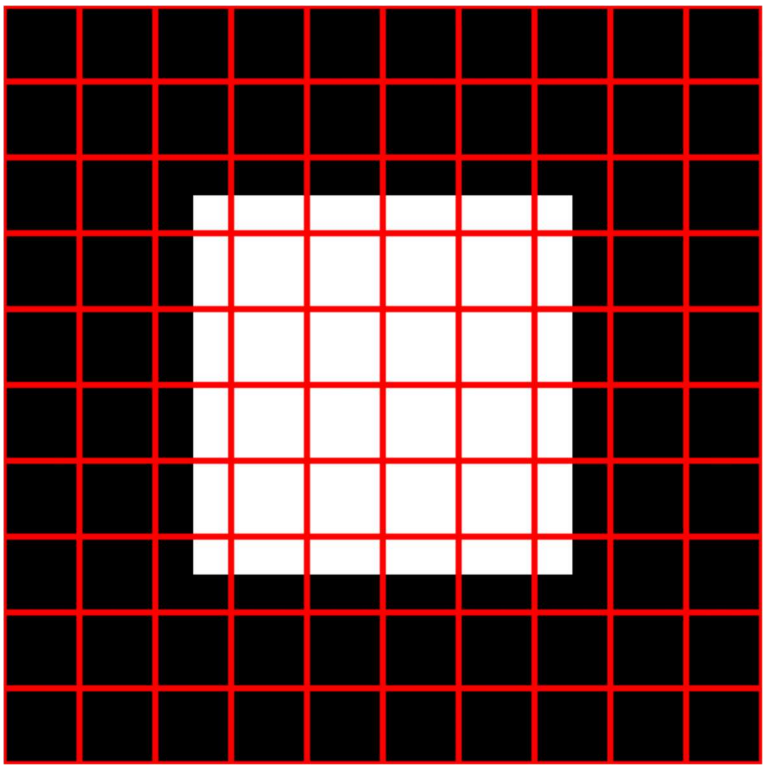
Desde o planejamento inicial deste trabalho houve a preocupação com a elevada quantidade de entradas que seriam utilizadas no processo de classificação pela RNA, assumindo que cada pixel da imagem seria uma entrada. Sabendo que o elevado número de entradas reduz consideravelmente a performance do treinamento da RNA, fez-se necessário desenvolver uma estratégia para a extração de características da imagem, de forma a reduzir o número de entradas e, consequentemente, otimizar o processamento.

A estratégia desenvolvida para solucionar o problema do dimensionamento do vetor de entradas se trata de um processo de “janelamento” da imagem. Isto é, a imagem foi dividida em janelas, onde todas as janelas contêm o mesmo número de pixels e o valor associado a cada

janela é dado pela média dos valores dos pixels contidos nela. Dessa maneira, o valor de cada janela passa a ser considerado uma entrada do classificador.

Assim, definindo-se uma janela quadrada, com laterais de 20 pixels, e levando-se em conta que cada imagem (área de detecção) possui 200 pixels em cada lateral, tem-se em uma imagem 100 janelas. Dessa maneira, a dimensão do vetor de entradas tem uma redução significativa – de 40.000 (200 pixels x 200 pixels) para apenas 100 entradas. A Figura 15 mostra um exemplo de como a imagem é dividida no processo de janelamento:

Figura 15 – Janelamento do padrão de imagem.



Fonte: Autoria própria.

Com o conjunto de dados gerado e uma estratégia de redução do dimensionamento do vetor de entradas definida, prosseguiu-se para o treinamento do PMC apresentando os dados de entrada e suas respectivas saídas desejadas.

O conjunto de dados foi dividido em um conjunto para o treinamento e outro para o teste e validação do classificador. As características da RNA, tais como número de camadas, quantidade de neurônios por camada e erro mínimo, foram definidos empiricamente e ajustados algumas vezes, por tentativa e erro.

A cada ajuste nas configurações realizado, a PMC foi treinada novamente e foi realizado um novo procedimento de teste, utilizando os mesmos conjuntos de dados separados previamente. Então, avaliou-se a taxa de acerto do classificador e sua performance após os ajustes realizados. Os resultados são apresentados no capítulo a seguir.

5 RESULTADOS

O script a seguir foi desenvolvido no software MATLAB para gerar o conjunto de imagens. No total, foram geradas 112.651 imagens.

```
clc; clear all; close all;

%% Dados
jan = 20; % Tamanho da Janela
cont = 1;

tic;
for z = -5:5
    for y = -(50-z):(50-z)
        for x = -(50-z):(50-z)
            Dados(cont, :) = [janelamento(deslocaPadrao(x, y, z), ...
                jan, 1) sinal(x) sinal(y) sinal(z)];
            cont = cont + 1;
        end
    end
end
tempo = toc;
```

Note que o deslocamento máximo considerado para o eixo z possui módulo 5, enquanto que o deslocamento máximo para os eixos x e y possui módulo 50. Além disso, é possível notar a presença de três funções no script: deslocaPadrao, janelamento e sinal.

A função deslocaPadrao, apresentada no Anexo I do presente trabalho, foi desenvolvida para gerar o padrão de imagem e deslocá-lo conforme seus atributos de entrada, os deslocamentos em x, y e z. Esta função retorna uma matriz, com 200 linhas e 200 colunas, contendo os valores dos respectivos pixels da imagem deslocada.

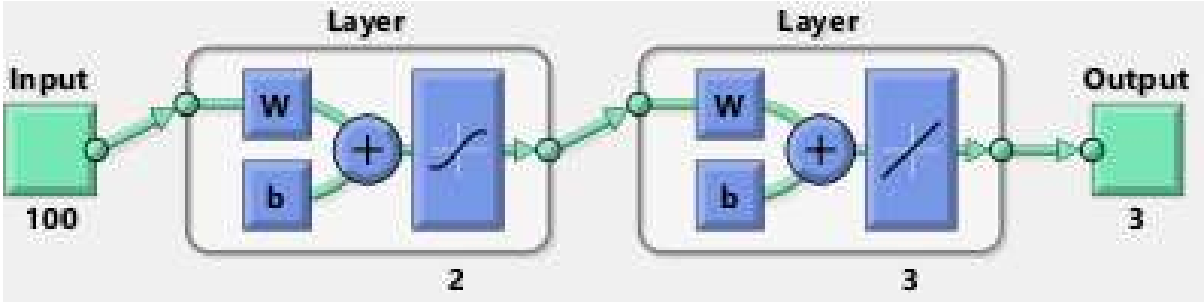
A função janelamento, apresentada no Anexo II, é responsável por implementar a estratégia de janelamento na imagem, de forma a reduzir o número de dados de entrada (dimensão do vetor de entradas). Esta função retorna um vetor – com dimensão 1 x J, onde J é número total de janelas – contendo os valores atribuídos a cada janela e possui três atributos de entrada: a matriz que representa a imagem, o tamanho da janela e a possibilidade de normalizar o valor de cada janela em valores de 0 a 1. Para este trabalho, foi adotada uma janela de tamanho 20 x 20 pixels. Deste modo, foi possível reduzir o número de entradas de 40.000 – considerando cada pixel da área de detecção como uma entrada – para apenas 100 entradas.

A função sinal, apresentada no Anexo III, retorna o sentido do deslocamento em determinado eixo e, portanto, apresenta a saída desejada, necessária para o treinamento supervisionado do classificador. Para deslocamentos negativos, isto é, contra o sentido de orientação do eixo em questão, o valor retornado é -1. Para deslocamentos positivos, o valor retornado é 1. Quando não há deslocamento no eixo em questão, o valor retornado é 0.

O conjunto de dados foi dividido em um grupo para treinamento, contendo 60% do total de amostras (67.590 amostras), e um grupo de teste, contendo os restantes 40% das amostras (45.061 amostras). Os dados foram divididos em grupos diferentes e de forma intercalada para atender a necessidade de o treinamento conter amostras onde os limites das classes estão inclusos, para maior eficácia da generalização do aprendizado. Dessa forma, prosseguiu-se para a etapa de modelagem do PMC.

As configurações do classificador foram definidas empiricamente. A priori, definiu-se um classificador com a estrutura mais simples da arquitetura multicamadas, ilustrado na Figura 16, contendo apenas uma camada oculta. Inicialmente, definiu-se 02 (dois) neurônios para esta camada.

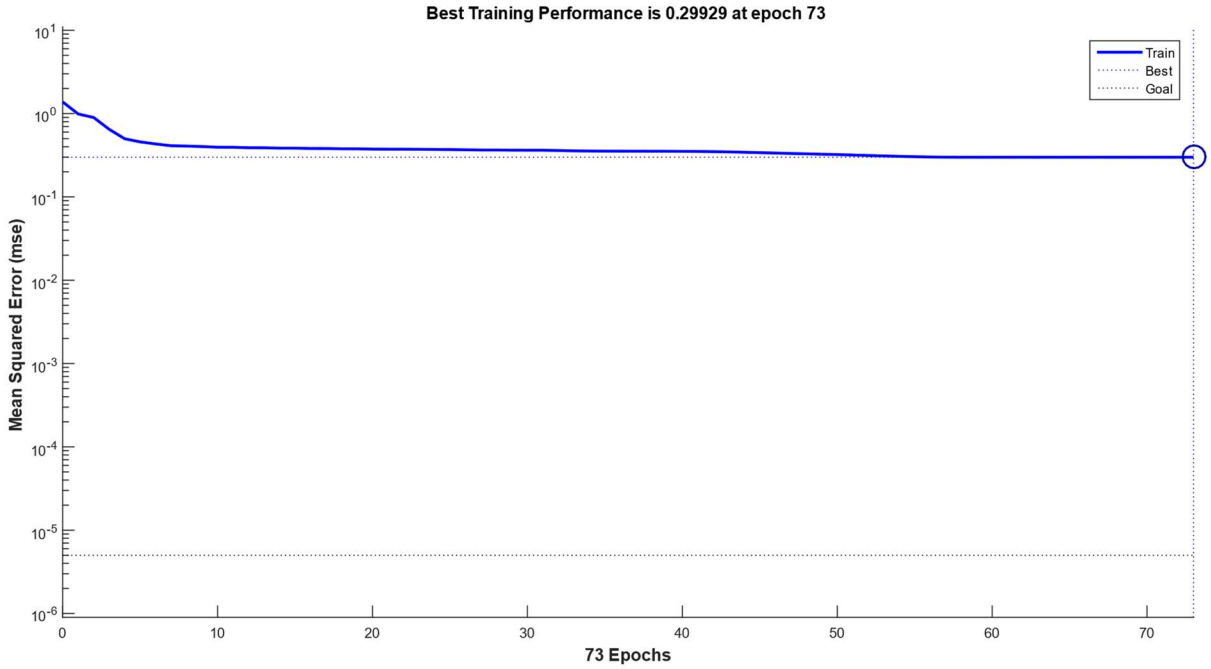
Figura 16 – Classificador 1.



Fonte: Autoria própria.

Com esta configuração, o treinamento foi interrompido na época 73, quando se atingiu o critério de parada do gradiente mínimo do erro. Portanto, não foi possível atingir o erro final desejado que convergiu para aproximadamente 0,299 por volta da época 5. A Figura 17 apresenta o comportamento do erro ao passar das épocas de treinamento:

Figura 17 – Erro final do Classificador 1.

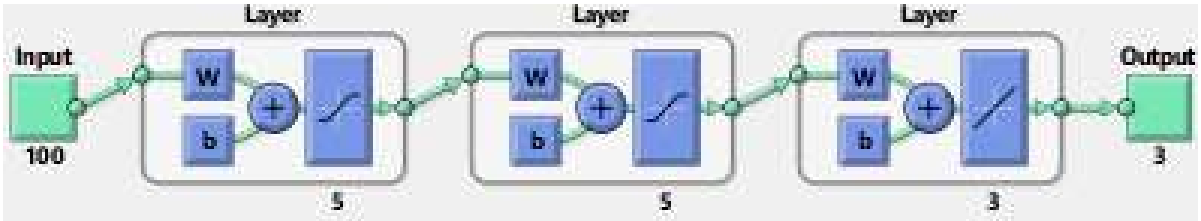


Fonte: Autoria própria.

Após a realização da validação, verificou-se que de fato a rede não foi capaz de classificar os padrões corretamente. A rede foi capaz de classificar corretamente as saídas x e y de todas as amostras de dados, porém houve inconsistência na grande maioria das amostras classificadas na saída z.

Então, decidiu-se aumentar o número de camadas ocultas com o intuito de aumentar a capacidade de processamento da rede de modo a atingir o erro desejado. Assim, o número de camadas aumentou de 01 (uma) para 02 (duas), com cada uma contendo 5 neurônios. A nova configuração do classificador é mostrada na Figura 18.

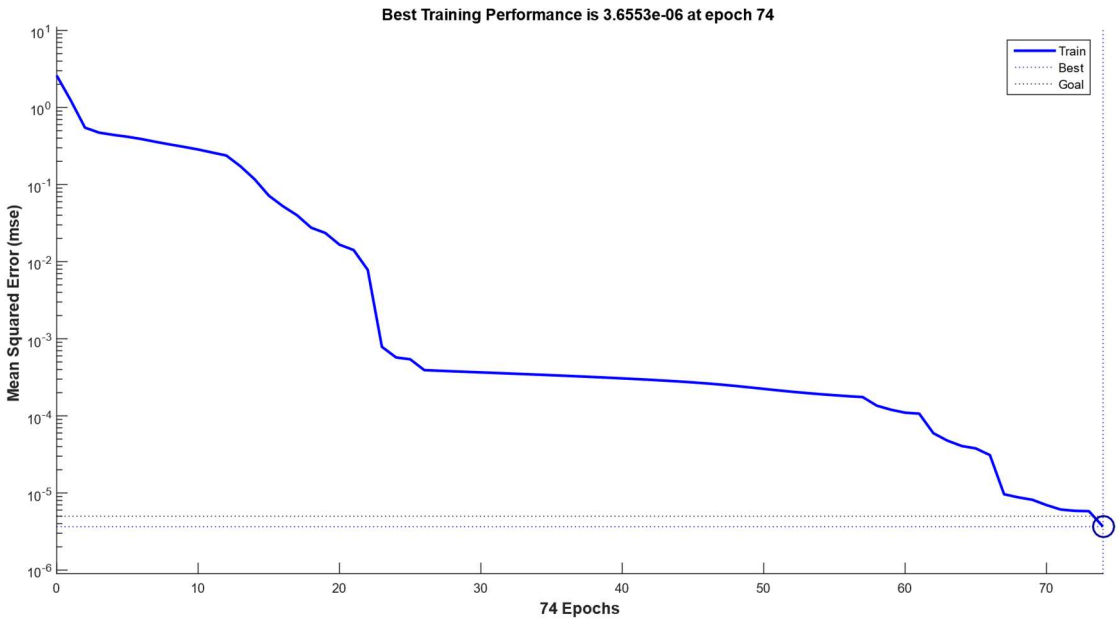
Figura 18 – Classificador 2.



Fonte: Autoria própria.

De fato, com a nova configuração a rede foi capaz de atingir o erro mínimo desejado na época 74, conforme pode ser observado no gráfico do erro quadrático de treinamento (Figura 19).

Figura 19 – Erro final do Classificador 2.

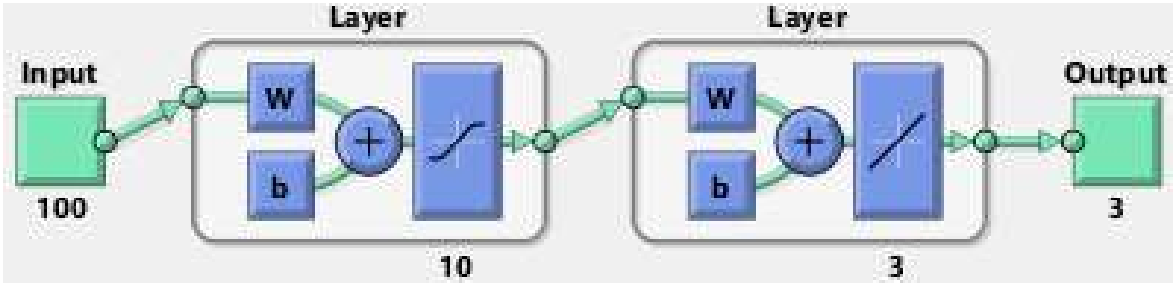


Fonte: Autoria própria.

Na etapa de validação, verificou-se que o fato de a rede ter convergido para o erro mínimo desejado na etapa de aprendizado contribuiu para um desvio padrão satisfatório entre a saída desejada e a saída real na classificação das amostras de teste. De fato, com esta configuração o classificador foi capaz de acertar 99,95% das amostras, errando a classificação de 23 amostras de um total de 45.061. Do total de erros, 5 amostras foram classificadas com a saída X divergente do desejado enquanto 18 amostras foram classificadas com a saída Y divergente do desejado.

Mesmo obtendo-se um resultado satisfatório, decidiu-se testar uma terceira configuração para o classificador. Desse modo, utilizou-se a configuração contendo apenas uma camada oculta, que passou a contar com um total de 10 (dez) neurônios. Assim, prosseguiu-se novamente para as etapas de aprendizado e teste do novo classificador. A Figura 20 mostra a configuração da nova arquitetura do classificador.

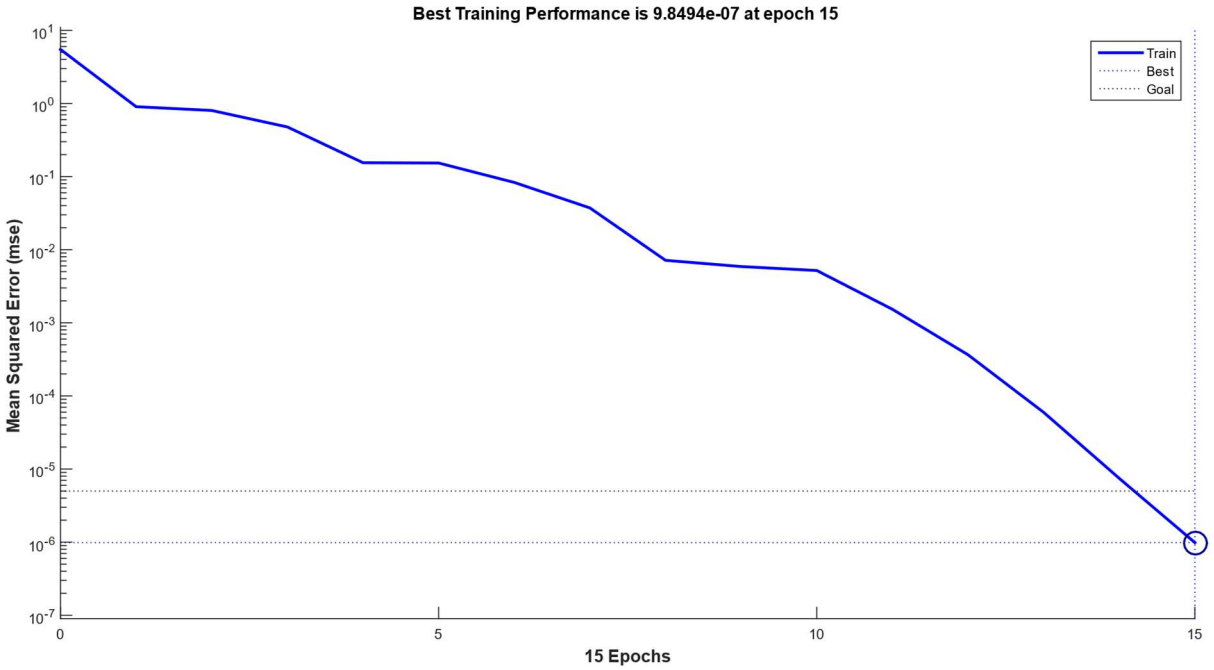
Figura 20 – Classificador 3.



Fonte: Autoria própria.

O aumento no número de neurônios da camada oculta resultou na convergência do erro para além do mínimo desejado, atingindo um erro em torno de 10^{-6}, conforme apresentado na Figura 21, que foi atingido na época 15.

Figura 21 – Erro final do Classificador 3.



Fonte: Autoria própria.

De fato, ao checar a classificação das amostras, verificou-se que com esta configuração 100% das amostras haviam sido classificadas corretamente.

Em resumo, a Tabela 1 mostra os parâmetros das três configurações utilizadas, enquanto a Tabela 2 mostra os respectivos resultados obtidos:

Tabela 1 – Parâmetros utilizados nos classificadores.

Classificador	**Neurônios 1º Camada Oculta**	**Neurônios 2º Camada Oculta**	**Erro Desejado**	**Qtd. Máx. de Épocas de Treino**	**Taxa de Aprendizado**
1	2	-	$5x10^{-6}$	1000	$5x10^{-3}$
2	5	5	$5x10^{-6}$	1000	$5x10^{-3}$
3	10	-	$5x10^{-6}$	1000	$5x10^{-3}$

Fonte: Autoria própria.

Tabela 2 – Resultados obtidos após os testes.

Classificador	**Tempo de Treino**	**Erro Atingido**	**Época Atingida**	**Critério de Parada**	**Quantidade de Erros**	**Taxa de Acerto**
1	2min 56s	0,299	73	Gradiente	45.061	0%
2	7min 20s	$3,66x10^{-6}$	74	Erro	23	99,95%
3	3min 42s	$9,85x10^{-7}$	15	Erro	0	100%

Fonte: Autoria própria.

6 CONSIDERAÇÕES FINAIS

As simulações desenvolvidas durante este trabalho retornaram resultados totalmente satisfatórios, comprovando a capacidade e funcionalidade de um sensor de força multiaxial com princípio de detecção por processamento e análise de imagens.

Ainda que algumas aproximações e desconsiderações tenham sido feitas, tais como a captura de imagens ideais desconsiderando os efeitos de distorção na imagem e as etapas do processamento de imagens, acredita-se que as mesmas não interfiram no resultado uma vez que o classificador possui aprendizado generalizado, aprendendo os padrões por meio das amostras de dados apresentadas. Além disso, o ambiente de captura da imagem, isto é, a parte interna do sensor, poderá contar com um ambiente de captura controlado, contendo condições de iluminação ideais e outras características para garantir uma captura adequada.

O processo de Análise Digital da imagem, adotado nas simulações, se mostrou eficiente e relativamente simples, com a utilização da estratégia de janelamento da imagem. Tal estratégia possibilitou uma redução considerável do número de dados de entrada do classificador, sem diminuir sua eficácia, elevando assim a qualidade da performance do classificador.

Assim, foi possível atingir todos os objetivos deste trabalho, tendo simulado o funcionamento do sensor proposto e realizado sua validação. No entanto, esta foi apenas a etapa inicial do projeto. Para a futura continuidade deste projeto, visa-se:

- projetar e desenvolver a estrutura flexível;
- implementar a detecção do módulo da força aplicada e, assim, determinar a faixa de medição do sensor;
- realizar o treinamento do classificador com amostras de dados reais;
- implementar o sensor.

REFERÊNCIAS

AUGUSTO, Karen S. **Identificação automática do grau de maturação de pelotas de minério de ferro**. Dissertação (Mestrado em Engenharia) – Programa de Pós-Graduação em Engenharia de Materiais, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2012.

BAPTISTA, Rodrigo G. **Análise experimental e numérica de uma viga sujeita a flexão**. Monografia (Bacharelado em Engenharia Mecânica), Universidade Tecnológica Federal do Paraná, Cornélio Procópio, 2016.

BARBOSA, Flávio de S. et al. **Modelagem numérica e análise experimental aplicadas ao projeto de uma célula de carga**. Universidade Federal de Juiz de Fora, Juiz de Fora, 2009.

BERG, Alexandre C.; MULLER, Daniel N. **Reconhecimento de caracteres utilizando redes neurais**. Curso de Pós-Graduação em Ciência da Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 1993.

BEZERRA, Juliana. **Fases da revolução industrial**. Disponível em: <https://www.todamateria.com.br/fases-da-revolucao-industrial/> Acesso em: 16 ago. 2018.

BUENO, Regis C. **Detecção de contornos em imagens de padrões de escoamento bifásico com alta fração de vazio em experimentos de circulação natural com o uso de processamento inteligente**. Tese (Doutorado em Tecnologia Nuclear – Reatores), Instituto de Pesquisas Energéticas e Nucleares, Universidade de São Paulo, São Paulo, 2016.

CARVALHO, A. S.; CABEÇA, L. F; CARÓSIO, M. G. **Desenvolvimento de um RMN sensor unilateral para aplicações em agropecuária**. In: Simpósio Nacional de Instrumentação Agropecuária, São Carlos, 2014. Disponível em: < https://ainfo.cnptia.embrapa.br/digital/bitstream/item/115544/1/229siagro-2014-print01.pdf> Acesso em: 8 set. 2018.

CASTRO, Fernando C. C. **Reconhecimento e localização de padrões em imagens utilizando redes neurais artificiais como estimadores de correlação espectral**. Dissertação (Mestrado em Engenharia Elétrica) – Programa de Pós-Graduação em Engenharia Elétrica, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, 1995.

DUNN, William C. **Fundamentos de instrumentação industrial e controle de processos**. Tradução de Fernando Lessa Tofoli. Editora Bookman: Porto Alegre, 2013.

EMBRAPA. **Desenvolvimento de sensores de RMN, portáteis e de baixo custo, para a análise não destrutiva de produtos agropecuários**. Brasília, 2014.

ERPEN, Luís R. C. **Reconhecimento de padrões em imagens por descritores de forma**. Dissertação (Mestrado em Ciência da Computação), Universidade Federal do Rio Grande do Sul, 2004.

GOMES, Otavio da F. M. **Microscopia co-localizada: novas possibilidades na caracterização de minérios**. Tese (Doutor em Engenharia Metalúrgica e de Materiais) – Programa de Pós-Graduação em Engenharia Metalúrgica, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2007.

GOMES, O. F. M. Processamento e Análise de Imagens Aplicados à Caracterização Automática de Materiais. Dissertação (Mestrado em Ciência de Materiais e Metalurgia), Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2001.

GONZALEZ, Rafael C.; WOODS, Richard E. **Processamento de imagens digitais**. Tradução de Cristina Yamagami e Leonardo Piamonte. 3. ed. São Paulo: Pearson Prentice Hall, 2010.

HAYKIN, Simon. **Redes neurais: princípios e prática**. Tradução de Paulo Martins Engel. 2.ed. Porto Alegre: Bookman, 2001.

HOFFMANN, Karl. An introduction to stress analysis and transducer design using strain gauges.

NISHIDA, Waleska. **Uma rede artificial para classificação de imagens multiespectrais de sensoriamento remoto**. Dissertação (Mestrado em Engenharia de Produção), Universidade Federal de Santa Catarina, 1998.

OGATA, Katsuhiko. Engenharia de Controle Moderno. Tradução de Bernardo Severo. 3.ed. Rio de Janeiro: LTC, 2000.

OMEGA. **Célula de carga: saiba mais sobre células de carga e tipos**. Disponível em: <https://br.omega.com/prodinfo/celulas-de-carga.html> Acesso em: 18 ago. 2018.

PERASSO, Valeria. O que é a 4ª revolução industrial – e como ela deve afetar nossas vidas. BBC News Brasil, São Paulo, 2016. Disponível em: < https://www.bbc.com/portuguese/geral-37658309> Acesso em: 19 ago. 2018.

RIBEIRO, Sergio S. **Extração de características de imagens aplicada a detecção de grãos ardidos de milho**. Dissertação (Mestrado em Computação Aplicada) – Programa de Pós-Graduação em Computação Aplicada, Universidade Estadual de Ponta Grossa, Ponta Grossa, 2015.

SCHMIDT, Stefan. **Como uma célula de carga realmente trabalha**. HBM, São Paulo. Disponível em: <https://www.hbm.com/pt/6768/como-uma-celula-de-carga-trabalha/> Acesso em: 18 ago. 2018.

SILVEIRA, Leonardo; LIMA, Weldson Q. **Um breve histórico conceitual da Automação Industrial e Redes para Automação Industrial**. Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal do Rio Grande do Norte, 2003.

SILVA, Aline F. **Estudo, desenvolvimento e concepção de uma célula de carga**. Monografia – Graduação em Engenharia Mecânica, Universidade Estadual Paulista, Guaratinguetá, 2012.

SILVA, Ivan N.; SPATTI, Danilo H.; FLAUZINO, Rogério A. **Redes neurais artificiais: para engenharia e ciências aplicadas**. São Paulo: Artilber, 2010.

SILVA, Renato R. **Reconhecimento de imagens digitais utilizando redes neurais**. Monografia – Graduação em Ciência da Computação, Universidade Federal de Lavras, Lavras, 2005.

STANGE, Renata L.; NETO, João J. **Reconhecimento de padrões em classificadores – comparação de técnicas e aplicações**. In: Quarto Workshop de Tecnologia Adaptativa - WTA 2010. São Paulo, 2010.

TIGGEMANN, L. et al. **Desenvolvimento de sensores de baixo custo para detecção de aromas alimentícios comerciais**. In: XX Congresso Brasileiro de Engenharia Química, Florianópolis, 2014. Disponível em: < http://pdf.blucher.com.br.s3-sa-east-1.amazonaws.com/chemicalengineeringproceedings/cobeq2014/0957-22148-163601.pdf > Acesso em: 8 set. 2018.

ANEXO I - Função deslocaPadrao(x, y, z)

```
function [PATTERN, Pad] = deslocaPadrao(x, y, z)

    cx = abs(x); % Modulo do deslocamento em X
    cy = abs(y); % Modulo do deslocamento em Y
    xRes = 640;  % Largura total da imagem
    yRes = 480;  % Altura total da imagem
    xArea = 200; % Comprimento da área de detecção
    yArea = 200; % Altura da área de detecção
    pSize = 100;  % Tamanho do padrão

    if (x > (xArea-(pSize+(2*z)))/2 || y > (xArea-(pSize+(2*z)))/2 || z >
20)
        display('Error!');
    else
        P = ones(pSize+(2*z), pSize+(2*z));
        desloc = (xArea-(pSize+(2*z)))/2;
        A1 = [zeros(desloc); zeros(pSize+(2*z), desloc); zeros(desloc)];
        A2 = [zeros(desloc, pSize+(2*z)); P; zeros(desloc, pSize+(2*z))];
        A3 = [zeros(desloc); zeros(pSize+(2*z), desloc); zeros(desloc)];
        A = [A1 A2 A3];

        % Deslocamento em X
        while(cx > 0)
            if(x < 0)
                for i=1:yArea
                    for j=1:xArea
                        if (j == xArea)
                            A(i, j) = 0;
                        else
                            A(i, j) = A(i, j+1);
                        end
                    end
                end
            else
                for i=1:yArea
                    for j=xArea:-1:1
                        if(j == 1)
                            A(i, j) = 0;
                        else
                            A(i, j) = A(i, j-1);
                        end
                    end
                end
            end
            cx = cx - 1;
        end

        % Deslocamento em Y
        while(cy > 0)
            if(y < 0)
                for i=yArea:-1:1
                    for j=1:xArea
                        if (i == 1)
                            A(i, j) = 0;
                        else
                            A(i, j) = A(i-1, j);
                        end
```

```
                    end
                end
            else
                for i=1:yArea
                    for j=1:xArea
                        if(i == yArea)
                            A(i, j) = 0;
                        else
                            A(i, j) = A(i+1, j);
                        end
                    end
                end
            end
            cy = cy - 1;
        end

        % Retorno
        PATTERN = A;
        Pad = size(A);
        %PATTERN = [zeros((yRes-yArea)/2, xRes);
        %           zeros(yArea, (xRes-xArea)/2) A zeros(yArea, (xRes-
xArea)/2);
        %           zeros((yRes-yArea)/2, xRes)];
    end
end
```

ANEXO II – Função janelamento(A, jan, normalizar)

```
function X = janelamento(A, jan, normalizar)

    X = [];
    for my = 1:size(A, 1)/jan
        for mx = 1:size(A, 2)/jan
            M = A(((my-1)*jan)+1: my*jan, ((mx-1)*jan)+1: mx*jan);
        X = [X sum(sum(M))];
        end
    end
    if (normalizar)
        X = X/(jan*jan);
    end
end
```

ANEXO III – Função sinal(x)

```
function s = sinal(x)
    if(x == 0)
        s = 0;
    elseif (x > 0)
        s = 1;
    elseif (x < 0)
        s = -1;
    end
end
```

ANEXO IV – Script de treinamento e teste dos classificadores

```
close all; clear all; clc;

%% Perceptron Multicamadas

% Carregar dados de treinamento
load('Dados_treinamento2.mat');
T = Dados_treinamento2;
vet_entrada = T(:, 1:end-3)';
vet_desejado = T(:, end-2:end)';

% Inicialização da Rede
net = newff([minmax(vet_entrada)], [2 3],...
    {'tansig' 'purelin'},...
    'trainlm');

% Definição dos Parâmetros de Treinamento
net.trainParam.epochs = 1000;            %Número de Epocas
net.trainParam.goal = 5e-6;             %Erro final desejado
net.trainParam.lr = 0.005;               %Taxa de aprendizado
net.trainParam.show = 10;                %Refresh da tela (épocas)
net.trainParam.showCommandLine = true;  %Informações no Comand window
net.trainParam.showWindow = true;       %Mostra interface do treinamento
net.trainParam.time = inf;              %Tempo maximo de treinamento

% Treinamento da Rede
net = train(net, vet_entrada, vet_desejado);
%net            -> Variável que recebeu a Rede Neural
%vet_entrada    -> Vetor de entradas (conjunto de treinamento)
%vet_desejado   -> Vetor de saída desejado (conjunto de treinamento)


% Carregar dados de teste
load('Dados_teste2');
V = Dados_teste2;
vet_teste_entrada = V(:, 1:end-3)';
vet_teste_desejado = V(:, end-2:end)';

% Teste da Rede
vet_saida = sim(net, vet_teste_entrada);
%vet_saida          -> Vetor com os resultados de saída do PMC
%net                -> Variável que instanciou o PMC
%vet_teste_entrada  -> Vetor com dados de entrada para teste

erro_relativo = abs((vet_teste_desejado -
vet_saida)./(vet_teste_desejado));

figure(1)
hist(vet_teste_desejado' - vet_saida');
title('Histograma de Validação');
xlabel('Desvio');
ylabel('Padrões');
legend('X', 'Y', 'Z');


qtd_erros1 = 0; qtd_erros2 = 0; qtd_erros3 = 0;
```

```
erros1 = 0; erros2 = 0; erros3 = 0;

% Normalização das Saídas para 1, 0 e -1 e Comparação
for n = 1:size(vet_saida, 2)
    % Eixo X
    if (vet_saida(1, n) >= 0.95)
        vet_saida(1, n) = 1;
    elseif (abs(vet_saida(1, n)) <= 0.05)
        vet_saida(1, n) = 0;
    elseif (vet_saida(1, n) <= -0.95)
        vet_saida(1, n) = -1;
    end

    % Eixo Y
    if (vet_saida(2, n) >= 0.95)
        vet_saida(2, n) = 1;
    elseif (abs(vet_saida(2, n)) <= 0.05)
        vet_saida(2, n) = 0;
    elseif (vet_saida(2, n) <= -0.95)
        vet_saida(2, n) = -1;
    end

    % Eixo Z
    if (vet_saida(3, n) >= 0.95)
        vet_saida(3, n) = 1;
    elseif (abs(vet_saida(3, n)) <= 0.05)
        vet_saida(3, n) = 0;
    elseif (vet_saida(3, n) <= -0.95)
        vet_saida(3, n) = -1;
    end

    % Comparação
    if(vet_saida(1, n) ~= vet_teste_desejado(1, n))
        qtd_erros1 = qtd_erros1 + 1;
        erros1 = [erros1 n];
    end
    if(vet_saida(2, n) ~= vet_teste_desejado(2, n))
        qtd_erros2 = qtd_erros2 + 1;
        erros2 = [erros2 n];
    end
    if(vet_saida(3, n) ~= vet_teste_desejado(3, n))
        qtd_erros3 = qtd_erros3 + 1;
        erros3 = [erros3 n];
    end
end
```