Universidade Federal
de Campina Grande

Centro de Engenharia Elétrica e Informática
Curso de Graduação em Engenharia Elétrica

Daniel Xavier Silva

Detecção de Parafusos em Circuitos Eletrônicos Utilizando Processamanto de Imagens

Campina Grande, Paraíba
Setembro de 2017

Daniel Xavier Silva

Detecção de Parafusos em Circuitos Eletrônicos Utilizando Processamanto de Imagens

Trabalho de Conclusão de Curso apresentado à Coordenação do Curso de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Engenharia Elétrica.

Universidade Federal de Campina Grande - UFCG

Centro de Engenharia Elétrica e Informática - CEEI

Departamento de Engenharia Elétrica - DEE

Orientador: Edmar Candeia Gurjão

Campina Grande, Paraíba

Agosto - 2017

Daniel Xavier Silva
Detecção de Parafusos em Circuitos Eletrônicos Utilizando Processamanto de Imagens/ Daniel Xavier Silva . – Campina Grande, Paraíba, Agosto - 2017-
54 p. : il. (algumas color.) ; 30 cm.

Orientador: Edmar Candeia Gurjão

Trabalho de Conclusão de Curso – Universidade Federal de Campina Grande - UFCG
Centro de Engenharia Elétrica e Informática - CEEI
Departamento de Engenharia Elétrica - DEE, Agosto - 2017.

1. Processamento Digital de Imagens. 2. Reconhecimento de Padrões. I. Universidade Federal de Campina Grande. II. Departamento de Engenharia Elétrica III. Detecção de Parafusos em Circuitos Eletrônicos Utilizando Processamanto de Imagens.

Daniel Xavier Silva

Detecção de Parafusos em Circuitos Eletrônicos Utilizando Processamanto de Imagens

Trabalho de Conclusão de Curso apresentado à Coordenação do Curso de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Engenharia Elétrica.

Trabalho aprovado. Campina Grande, Paraíba,

Edmar Candeia Gurjão
Universidade Federal de Campina Grande
Orientador

Luciana Ribeiro Veloso
Universidade Federal de Campina Grande
Avaliador

Campina Grande, Paraíba
Agosto - 2017

Agradecimentos

Agradeço a Deus, Aquele que tem controle sobre todas as coisas e que nos possibilitou estar onde estamos.

Agradeço a minha família pelo suporte nesta jornada, em especial aos meus pais Pedro e Sueli, que sempre fizeram todo o possível para fornecer todo o apoio possível para que eu pudesse chegar nesta etapa.

Agradeço ao meu orientador Edmar, por ter me feito ganhar cada dia em que nos reuníamos para discutir o trabalho.

"Pensamos em demasia e sentimos bem pouco. Mais do que de máquinas, precisamos de humanidade. Mais do que de inteligência, precisamos de afeição e doçura. Sem essas virtudes, a vida será de violência e tudo será perdido." (Charles Chaplin)

Resumo

Há uma vasta gama de aplicações na indústria no que diz respeito a automatização de processos, em que a informação extraída de imagens faça parte de um conjunto de sensores que controlam máquinas. Neste trabalho, foi proposto e testado um sistema de detecção de parafusos utilizando processamento digital de imagens. Os componentes do sistema estão descritos, bem como os parâmetros que os caracterizam, seguido pelos resultados. As imagens obtidas para teste e treinamento de um classificador foram obtidas em um ambiente industrial.

Palavras-chave: Processamento Digital de Imagens. Detecção de Parafusos. Reconhecimento de Padrões

Abstract

There are a wide range of applications in the industry when it comes to process automation, where information extracted from images is part of a set of sensors that control machines. In this work, a screw detection system was proposed and tested using digital image processing. The components of the system are described, as well as the parameters that characterize them, followed by the results. The images obtained for the tests and training of a classifier were obtained in an industrial environment.

Keywords: Digital Image Processing. Screw Detection. Pattern Recognition.

Lista de ilustrações

Lista de tabelas

Lista de símbolos

∇	Operador Gradiente
w_i	i-ésimo peso em um Perceptron numa Rede Neural
σ	parametro da Função Gaussiana
G	Máscara Gaussiana
G'	Aproximação da Derivada da Máscara Gaussiana
θ	Ângulo de direção
x_i	I-ésimo elemento do vetor de entrada em um classificador
w_{qp}	P-ésimo peso do q-ésimo Perceptron Rede Neural na última camada de uma rede neural perceptron multicamada
w_{pj}	P-ésimo peso do j-ésimo Perceptron Rede Neural na penúltima camada de uma rede neural perceptron multicamada
α	Taxa de aprendizado
Δ	Indicador de Variação
$\frac{\partial}{\partial}$	Operador derivada parcial

Sumário

1 Introdução

Processamento de Imagens é uma área que vem sendo desenvolvida desde os anos 20 do século passado e tem duas grandes áreas de aplicação: melhoramento/construção de imagens para interpretação humana e processamento de dados para armazenamento, transmissão e representação para percepção autônoma de máquinas. (3)

Uma aplicação, que irá ser desenvolvida nesse trabalho, é a localização de parafusos através de imagens para um futuro sistema de automação de retirada de parafusos de produtos.

O documento está organizado em 5 capítulos, O segundo capítulo trata-se da fundamentação teórica dos elementos constituíntes do sistema. O terceiro capítulo trata-se da metodologia, o diagrama de blocos do sistema e dos dados obtidos. O quarto capítulo são mostrados os resultados e análises. O quinto capítulo mostra as considerações finais e sugestões para futuros trabalhos neste mesmo tema. Após a conclusão, tem-se as referências bibliográficas e os anexos, com os scripts utilizados para implementação.

2 Fundamentação Teórica

2.1 Pré-Processamento - Filtragem e Detecção de Bordas

O algoritmo de detecção de bordas utilizado no trabalho é o de Canny, que é considerado um dos melhores já propostos e foi o que melhor trouxe resultados para o trabalho dentre os testados, que foram além deste, o de Sobel e Prewitt.

O processo de detecção de Canny pode ser sumarizado nos seguintes passos: suavização da imagem com o filtro Gaussiano, computação da magnitude e direção de gradientes, aplicação da supressão nonmaxima, definçã um limiar para que as bordas da imagem estejam acima deste e suprimir bordas fracas que não estão conectadas em bordas fortes. (3)

A filtragem passa-baixa nesse sistema tem-se como motivação o melhoramento da relação sinal-ruído (SNR) na imagem, implicando na diminuição da quantidade de bordas geradas por ruídos no processo de detecção de bordas. As componentes de frequência de uma imagem geralmente são concentrados na região de baixas frequências e do ruído em todas as frequências, então a estratégia da filtragem é diminuir a amplitude da potência do ruído em altas frequências.

O filtro gaussiano tem como expressão matemãtica no domínio espacial a equação abaixo:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma}} \tag{2.1}$$

A imagem filtrada é obtida pela convolução entre a própria imagem e o filtro.

Nesse trabalho, a filtragem é feita no domínio da frequência, o que equivale a aplicação da FFT bidimensional, uma multiplicação pela transformada discreta de Fourier de G(x,y) e a transformada inversa do resultado. A transformada de Fourier do filtro é dado por:

$$G(u,v) = e^{-2\pi^2\sigma^2(u^2+v^2)} \tag{2.2}$$

O filtro gaussiano tem como parâmetro σ^2, que tem analogia com a variância da variável aleatória gaussiana. Quanto maior o σ^2, mais larga será a curva no domínio espacial e mais estreita será no domínio da frequência, assim diminuindo a frequência de passagem do filtro. Acerca do ajuste desse parâmetro, se tem valor muito baixo, este não irá atenuar o ruído de forma satisfatória, se muito alto, poderá comprometer informações relevantes das regiões de bordas, comprometendo o processo de detecção. Levando isso em consideração, foi testado nas imagens de banco de dados a filtragem com diferentes

parâmetros para ver quais se ajustavam melhor. A Imagem da resposta em frequência do filtro gaussiano e como é o seu comportamento para diferentes valores de σ está na figura 1.

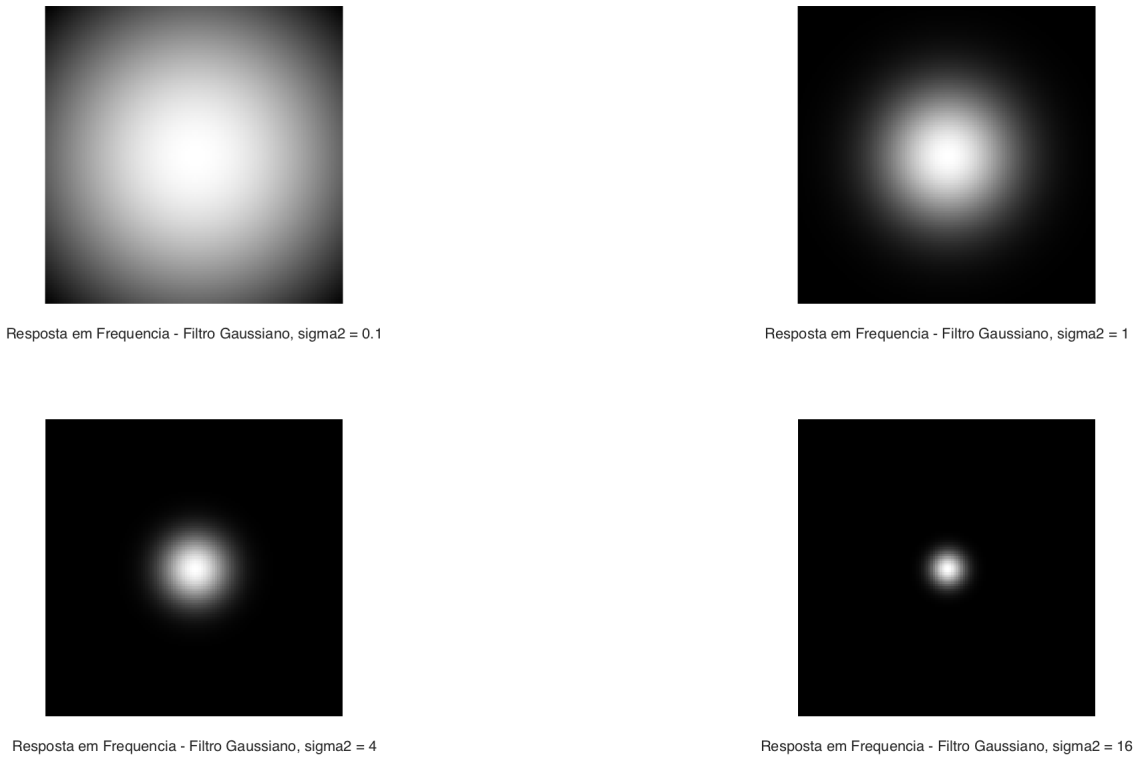


Figura 1 – Respostas em Frequência de Filtros Gaussianos para Diferentes valores de σ^2

Acerca do cálculo do gradiente da imagem, deseja-se adaptar o caso contínuo para o discreto. O gradiente para $f(x, y)$ - que representa uma imagem - pode ser descrito por duas componentes: a derivada na direção x representada por $(\frac{\partial f(x,y)}{\partial x})$ e a derivada na direção y, representada por $(\frac{\partial f(x,y)}{\partial y})$. O seu módulo é dado por $((\frac{\partial f(x,y)}{\partial x})^2 + (\frac{\partial f(x,y)}{\partial y})^2)^{\frac{1}{2}}$ e direção, $\arctan \frac{\frac{\partial f(x,y)}{\partial y}}{\frac{\partial f(x,y)}{\partial x}}$.

Uma adaptação disso para o caso discreto, o operador derivada direcional em um pixel pode ser representado como a diferença entre este pixel e o que está antes ou após este na direção especificada, no caso da direção x, o pixel imediatamente a esquerda, no caso da direção y, um pixel abaixo, mais ainda essa operação pode ser pensada como uma filtragem, no caso da derivada direcional na direção X, o kernel a ser utilizado pode ser $[-1 \;\; 1]$ e no caso da direção Y, o kernel seria $[-1 \;\; 1]^T$.

O detector de bordas de Canny consiste em criar duas máscaras: uma máscara Gaussiana G unidimensional para convoluir com a imagem a fim de filtrá-la e outra máscara G' para computar uma aproximação do gradiente. Acerca da máscara G, no MATLAB®(MATrix LABoratory), sua dimensão é de oito vezes o parâmetro σ e busca achar valores discretos e igualmente espaçados no intervalo $[-3.5\sigma, 3.5\sigma]$. Após isso, uma

máscara G' que computa a derivada dessa máscara Gaussiana que é aproximada pela equação 2.3. Observa-se que a expressão para definir o gradiente é diferente do que foi descrito no paragráfo anterior, na verdade há várias expressões para aproximar o gradiente e essa expressão utilizada mantém a coerência na operações feitas. (5)

$$G' = G * [-1/2 \quad 0 \quad 1/2] \tag{2.3}$$

As operações para se obter a aproximação dos gradientes filtrados nas direções X e Y são obtidas, considerando I(x,y) a imagem, através da equações 2.4 e 2.5

$$I_x = G' * I(x, y) \tag{2.4}$$

$$I_y = G'^T * I(x, y) \tag{2.5}$$

Computa-se a direção e magnitude dos gradientes, a partir disso vem o passo final do detector, que é o chamado de supressão nonmaxima. Usa-se um valor limiar, em que a partir dele, todos os gradientes abaixo encontrados sejam zerados, após isso, o algoritmo busca os gradientes que sejam máximos locais na direção quando comparados com sua vizinhança. Após isso, começa as iterações, em que o algoritmo atribui o máximo local como um pixel de borda e, levando em consideração sua direção do gradiente, constrói-se um contorno em que o próximo pixel de borda é atribuindo saindo do anterior na direção do gradiente. Vale salientar que os pixels vizinhos que não não estão sendo apontados pela direção deste gradiente e que não são tampouco máximos locais são zerados. Vale salientar que nem sempre o máximo local está em um píxel que estaria em um índice inteiro da matriz I(x,y) nem a direção do gradiente em cada pixel aponta diretamente para os pixels em sua volta, mas utilizando interpolação e atribuindo os pixels mais próximos destes pontos (x,y) que não podem ser indexados como um ponto de borda é uma estratégia utilizada (5).

A imagem de bordas resultante é uma imagem lógica onde os pixels de valor 1 são as bordas detectadas.

2.2 Transformada Circular de Hough

Transformada Circular de Hough é uma técnica de extração de parâmetros utilizada em processamento de imagens para detecção de círculos.

De maneira geral, a transformada de Hough pode ser utilizada para detectar formas que podem ser parametrizadas, como por exemplo, uma linha, que pode ser parametrizadas

por dois parâmetros: (ρ, θ) e um círculo, que pode ser descrito por 3 parâmetros: centro (x,y) e raio R (6).

A Transformada Hough Circular pode ser descrita como uma função que recebe uma matriz logica de bordas de uma imagem e a faixa de valores de raios dos círculos a serem detectados e retorna os parâmetros dos círculos detectados na imagem.

A detecção de círculos via transformada Hough Circular é feita da seguinte maneira: detecta-se primeiro os pontos que são candidatos a centros dos círculos que estão contidos nas imagens e a partir deles estima-se os raios dos círculos parametrizados (6).

Pode-se definir uma matriz de acumulação, também chamada de acumulador, como uma matriz de mesma dimensão da matriz de bordas, em que cada elemento representa um candidato ao centro de um círculo localizado mesmo índice da imagem. (6)

O procedimento para que, a partir de pixel de borda, se encontre os possíveis candidatos a centro de círculos é o seguinte: se um pixel de borda está contido em um circulo, a direção do gradiente deste vetor é a mesma direção do vetor que liga o centro do círculo a este pixel. Assim, incrementa-se no acumulador todos os elementos que estão a uma distância mínima R1 e máxima R2 da borda no sentido oposto do gradiente. R1 e R2 sã a faixa dos valores mínimos e máximos dos possíveis círculos que serão encontrados na imagem. A imagem 2 mostra como funciona o processo para o circulo. A imagem mostra a matriz de bordas de um círculo e vetores que representam a direção do gradiente. O segmento de reta em destaque mostram os elementos da matriz acumuladora que serão incrementados no procedimento do algoritmo para os 3 pixels P1, P2 e P3. (6)

Na figura 3, vemos o procedimento feito para todos os pixels de bordas. Nota-se que há um aglomerado de pixels próximos ao centro dos possíveis círculos a serem detectados e um deles e em cada aglomeração, os máximos locais são escolhidos para ser os centros dos círculos (6) . Vale salientar que é possível que arcos de círculos e espirais formem uma região cujo o pico também pode ser detectado como um círculo e para essa aplicação, não é necessário levar em consideração.

2.3 Descrição - Assinatura

A assinatura é uma função que mapeia bordas de uma imagem em um vetor unidimensional que possa representá-la. A necessidade desse procedimento no trabalho é para que um vetor unidimensional que represente a imagem possa ser utilizado como entrada em um classificador. (3)

Para que a assinatura seja uma forma eficiente de descrever um objeto, é necessário que ela seja invariante a translação, rotação e escalonamento (3). O círculo previamente detectado é redimensionamento para uma matriz 128x128, assim fazendo o procedimento

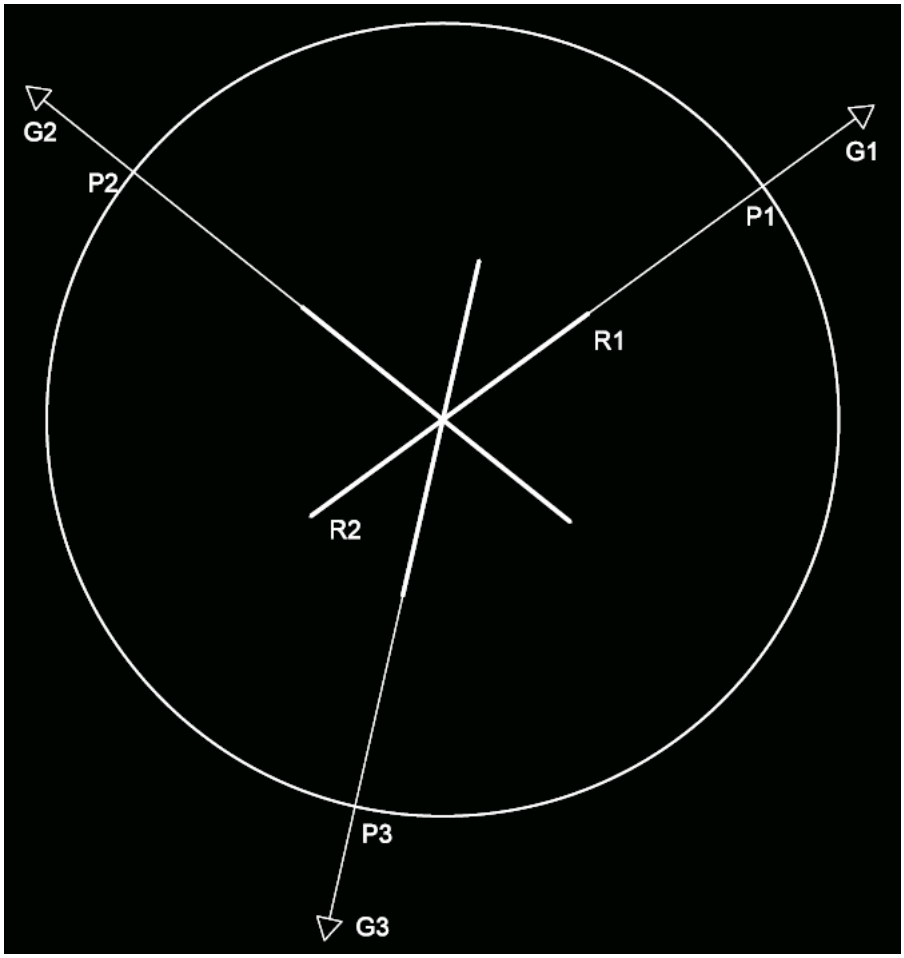


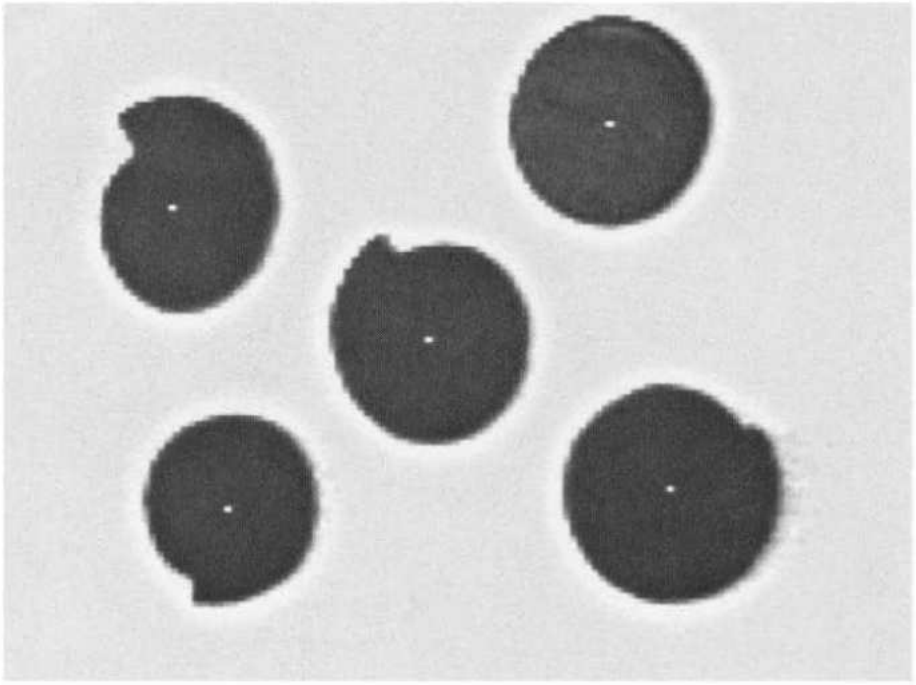
Figura 2 – Imagem de Bordas de um Círculo e o Vetor gradiente em 3 pixels (fora de escala)

para que a assinatura seja invariante a translação e escalonamento.

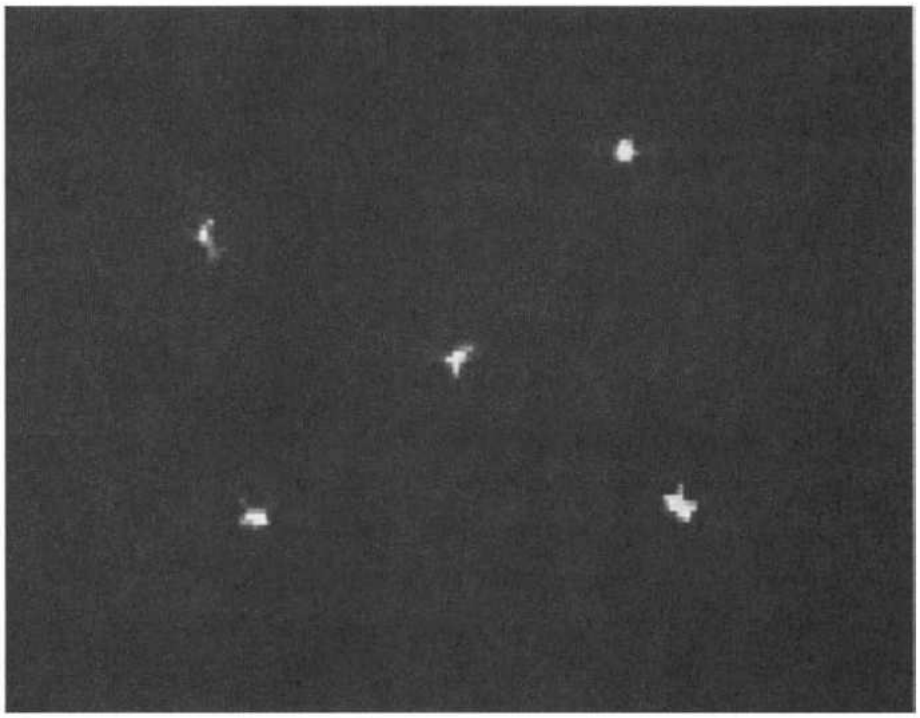
O algoritmo da assinatura utilizado neste trabalho é dado pela distância euclidiana d do centro do círculo até um ponto das bordas mais próximo em função de uma direção θ, que varia no sentido horário. No trabalho, o intervalo de discretização do ângulo θ é 30° , sendo assim, a dimensão do vetor unidimensional que representa a assinatura é 12. Foram testados valores de discretização para θ menores do que este, mas verificou-se que não há muita diferença na forma do vetor obtido.

A respeito da invariância à rotação, o procedimento feito em relação ao vetor obtido foi deslocar fazer o deslocamento circular dos elementos do vetor afim de que o índice que tem o elemento de valor máximo seja deslocado para o índice inicial.

O deslocamento parte da premissa de que o valor máximo é a distância máxima do centro do parafuso até uma das ranhuras. Dessa forma, fazendo esse deslocamento equivale a dizer que uma das ranhuras de um potencial parafuso irão estar na direção 0°. Um exemplo de uma imagem de bordas de um parafuso e sua assinatura é mostrado na figura 4.



(a) Imagem de Quatro Espirais e Um Círculo



(b) Matriz de Acumulação da figura em (a)

Figura 3 – Exemplo de Matriz de Acumulação obtida a partir de uma imagem (1)

2.4 Reconhecimento - Redes Neurais Artificiais

Redes neurais são uma subárea de aprendizagem de máquina que tem vastas aplicações: reconhecimento de padrões, classificação e predição baseada em dados. A

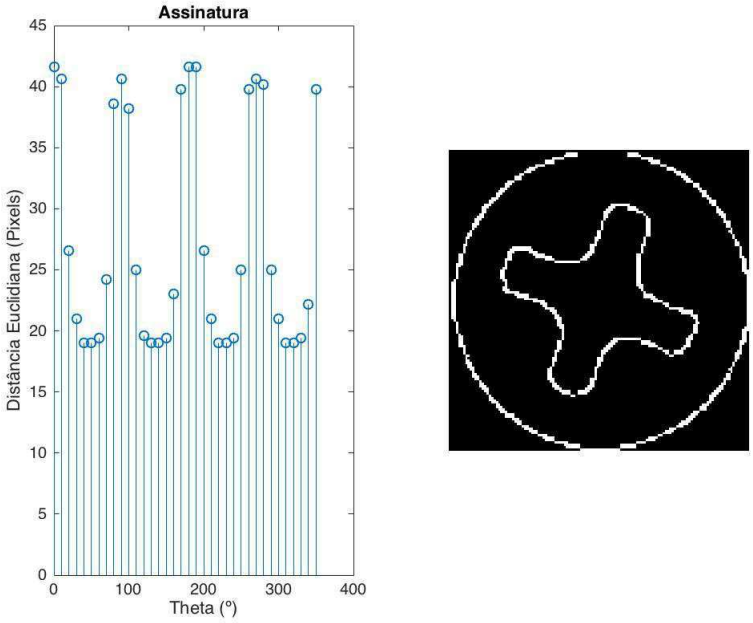


Figura 4 – Bordas de um Parafuso e sua Assinatura (2)

grande utilidade de redes neurais é que, além de resolver problemas de reconhecimento de padrões, há a possibilidade dela achar seus próprios parâmetros através de um processo chamado de treinamento. Neste trabalho, o tipo de rede neurais utilizadas é o perceptron multicamada.

No caso do Perceptron, o modelo de um recebe um vetor da forma $[x_1 \ x_2 \ x_3...x_n \ 1]^T$ na entrada, onde $x_i, i = 1, 2...n$ que é multiplicado pela direita e por $[w_1 \ w_2 \ w_3 \ ... \ w_n \ w_{n+1}]$, onde cada $w_1 \ w_2 \ w_3 \ ... \ w_n$ são chamados de pesos e w_{n+1} é chamado de elemento de ativação. O resultado dessa multiplicação é um escalar e a saída da rede é uma função desse escalar. Essa função é chamada de função de ativação. A representação de um Perceptron está Figura 5 (3).

A função mais simples que pode ser usada é a do tipo Limiar, mas conforme a teoria avança, há algoritmos de treino onde se faz necessário que a função de excitação seja diferenciável. Algumas funções comumente utilizadas como funções de ativação estão na Figura 6 .Entre essas funções, a sigmóide é uma das mais usadas e foi utilizada nesse trabalho como função de ativação. (3)

Um perceptron é apto a separar em classes que são linearmente separáveis. Por exemplo, no caso bidimensional, pode-se observar que, geometricamente falando, os 3 parâmetros $[w_1 \ w_2 \ w_3]$ compõem uma equação da reta, que após o treinamento, essa reta irá ser definida e o perceptron irá classificar os elementos da classe que estejam acima ou abaixo dela – analogamente, pode-se pensar o caso N-dimensional. em que os parâmetros compõem a equação de um hiperplano. Para tornar o sistema mais aplicável em casos

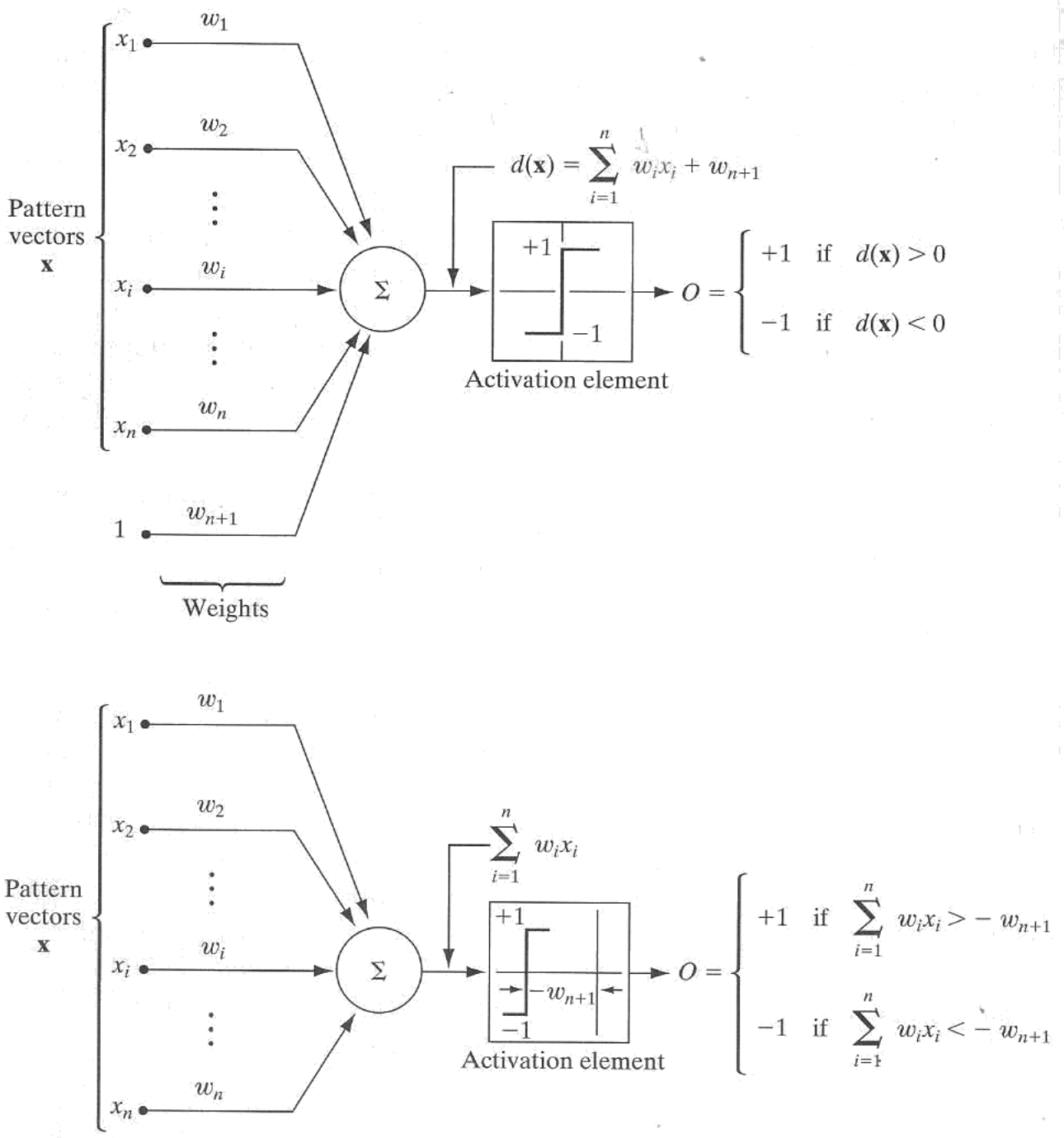


Figura 5 – Representação de um Perceptron. Fonte: (3)

.

reais, em que no geral, as classes não são linearmente separáveis, é necessário aumentar o número de perceptrons e o número de camadas – e aí se fundamenta o conceito de rede perceptron multicamada. A figura 7 mostra uma rede neural perceptron multicamadas genérica (3).

Em redes neurais perceptron multicamadas, define-se camada como um conjunto de perceptrons que tem as mesmas entradas em comum, que podem vir a partir do vetor

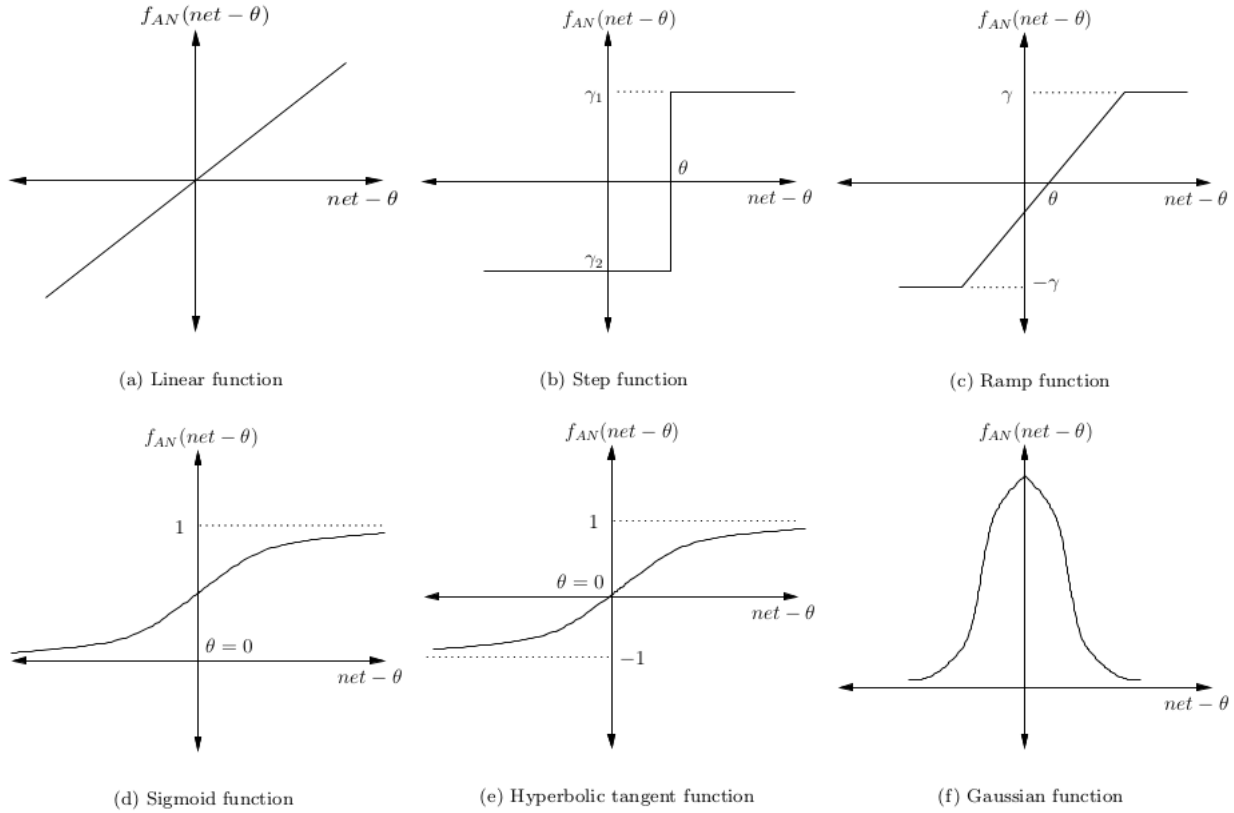


Figura 6 – Algumas Funções de Ativação. (4)

de entrada (input layer – camada de entrada), de outra camada e que vai para outra camada (hidden layer – camada escondida) e a camada final, que dá a informação das classificações (output layer – camada de saída). A representação de uma rede neural perceptron multicamadas está na Figura 7. (3)

Acerca das relações entre as camadas e dados, a quantidade de pesos da camada de entrada tem que ser a dimensão do vetor de entrada, a quantidade de pesos em cada perceptron das camadas escondidas e a de saída tem que ser a quantidade de perceptrons das camadas anteriores. Acerca da quantidade de perceptrons em cada camada, há apenas uma restrição: a quantidade de perceptrons da camada de saída é igual ao número de classes da rede.

Acerca do treinamento, o método utilizado é o da minimização da função erro quadrático via retroprogragação do gradiente.

A função erro quadrático (MSE) é definida por:

$$E_q = \sum_{q=1}^{N_q} (r_q - O_q)^2 \tag{2.6}$$

Onde r_q é a saída desejada, O_q é a saída calculada no perceptron q na camada de

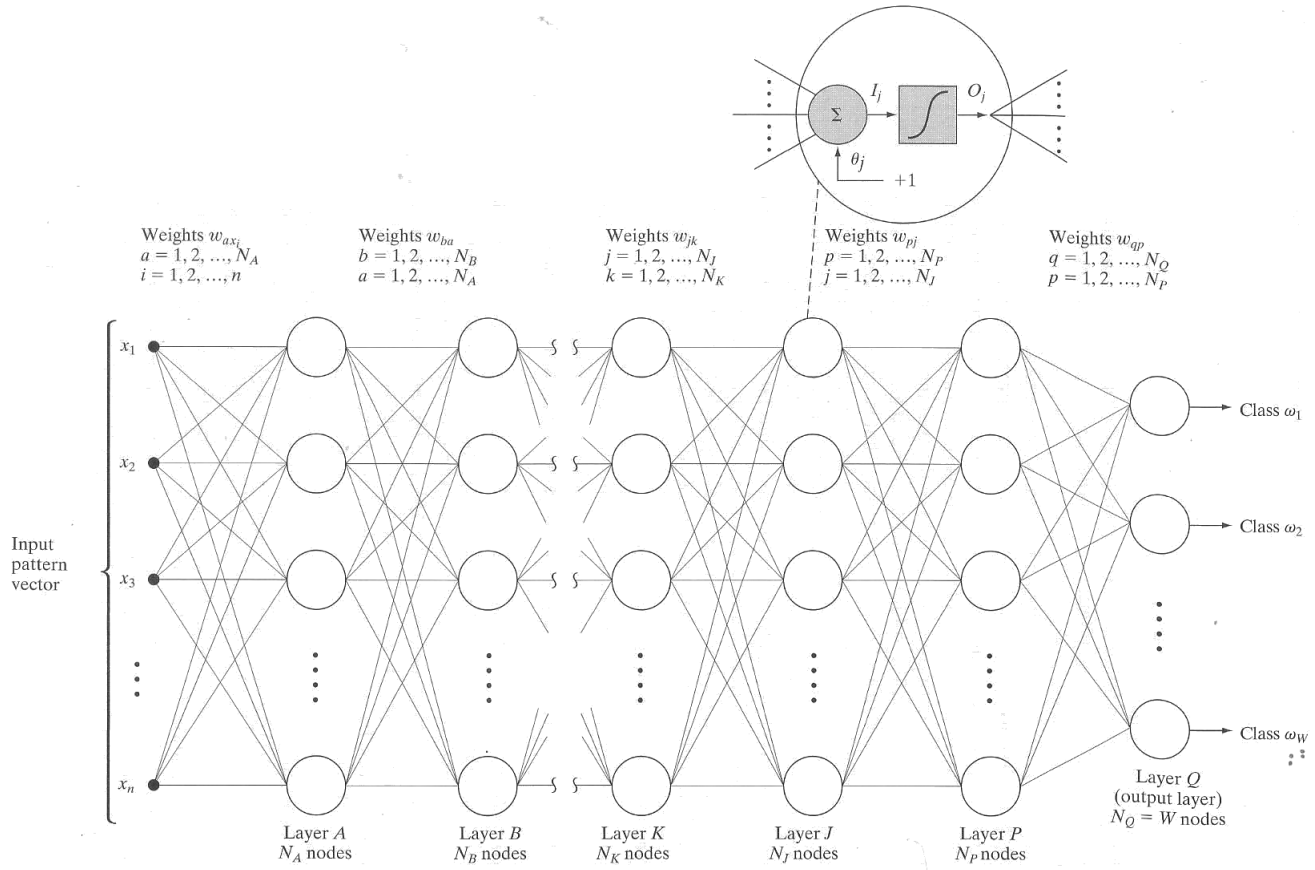


Figura 7 – Esquema do Perceptron Multicamada (3)

saída. A equação para corrigir os parâmetros é dada por:

$$w_{qp \quad atual} = w_{qp \quad anterior} - \Delta w_{qp} \tag{2.7}$$

Onde:

$$\Delta w_{qp} = -\alpha \frac{\partial E_q}{\partial w_{qp}} \tag{2.8}$$

O parâmetro α é chamado de taxa de aprendizado e que controla a taxa de convergência do método. Se muito pequeno, a convergência é lenta, se muito grande a solução pode divergir. Os valores típicos para α estão entre 0.1 e 1 (3). Ainda na equação 2.8, w_{qp} faz referencia ao p-ésimo peso do q-ésimo perceptron da última camada, conforme mostrado na Figura 7.

Deseja-se encontrar um valor para $\frac{\partial E_q}{\partial w_{qp}}$ que seja função de valores previamente conhecidos. É provado matematicamente que o valor de Δw_{qp} para o p-ésimo peso do q-ésimo perceptron na última camada pode ser expresso em função dos resultados obtidos na rede pela expressão: (1)

$$\Delta w_{qp} = -\alpha(r_q - O_q)h'_q(I_q)O_p \tag{2.9}$$

Onde $h'_q(I_q)$ é a derivada da função de ativação em I_q, sendo I_q a soma da multiplicação dos pesos atuais pelas saídas dos perceptrons da camada anterior.

Generalizando o resultado da equação 2.13 para a camada P, considerando a topologia mostrada na Figura 7 a equação 2.13 teria a seguinte forma:

$$\Delta w_{pj} = -\alpha(r_p - O_p)h'_p(I_p)O_j \tag{2.10}$$

Todos os valores da equação são conhecidos, exceto r_p, pois não se sabe a priori qual a resposta desejada de um perceptron que está em uma camada escondida. Porém, é demonstrado matematicamente que: (3)

$$(r_p - O_p) = \sum_{q=1}^{N_q}(r_q - O_q)h'_q(I_q)w_{qp} \tag{2.11}$$

Pode-se observar que na equação 2.11, que há as informações sobre os pesos, saídas e repostas desejadas da última camada que estão sendo usados para correção dos pesos da penúltima. Para a camada J, a formula para a obtenção dos termos de correção dos pesos é dada por: (3)

$$\Delta w_{jk} = -\alpha(r_j - O_j)h'_j(I_j)O_k \tag{2.12}$$

Onde:

$$(r_j - O_j) = \sum_{p=1}^{N_p}(r_p - O_p)h'_p(I_p)w_{pj} = \sum_{p=1}^{N_p}\sum_{q=1}^{N_q}(r_q - O_q)h'_q(I_q)w_{qp}h'_p(I_p)w_{pj} \tag{2.13}$$

A partir daí, observa-se que as obtenções das correções dos pesos podem-se propagar de forma recursiva, onde os termos de correção dos pesos da penúltima camada é função dos termos obtidos na última camada, a correção dos pesos na antepenúltima é função dos termos resultados na penúltima e assim sucessivamente até a camada de entrada.

No que diz respeito ao conjunto de dados a serem utilizados para treinamento e avaliação da rede, há três subconjuntos: treino, teste e validação. O conjunto de treino é o conjunto em que irá ser feito o treinamento da rede utilizando o treinamento via retropropagação do gradiente. O conjunto de validação é o conjunto em que cada elemento deste é aplicado na entrada na rede e o número de erros é contabilizado e armazenado para repetidos treinamentos. Após os primeiros treinamentos, a quantidade de erros

referentes aos dados de validação seja decrescente, mas a partir de uma quantidade de treino, a quantidade de erros começa a aumentar e é nesse ponto do treinamento em que a quantidade de erros da rede para com os dados de validação é mínimo, tem-se os valores finais dos pesos dos perceptrons da rede. Sobre os dados de teste, estes devem ser aplicados ao sistema ap 'os todo o processo de treinamento para avaliar a perfomance da rede (7).

O aumento do número de classificações erradas da rede a partir de uma certa quantidade de treinamentos da rede é chamado de sobre-ajuste (overfitting). Esse fenômeno descreve a situação em que a partir de um certo ponto no treinamento, os parâmetros da rede se ajustam excessivamente aos dados de treino, comprometendo a classificação correta de vetores na entrada (7).

3 Metodologia

O Sistema de processamento de imagens para detecção de parafusos apresenta o seguinte diagrama de blocos:

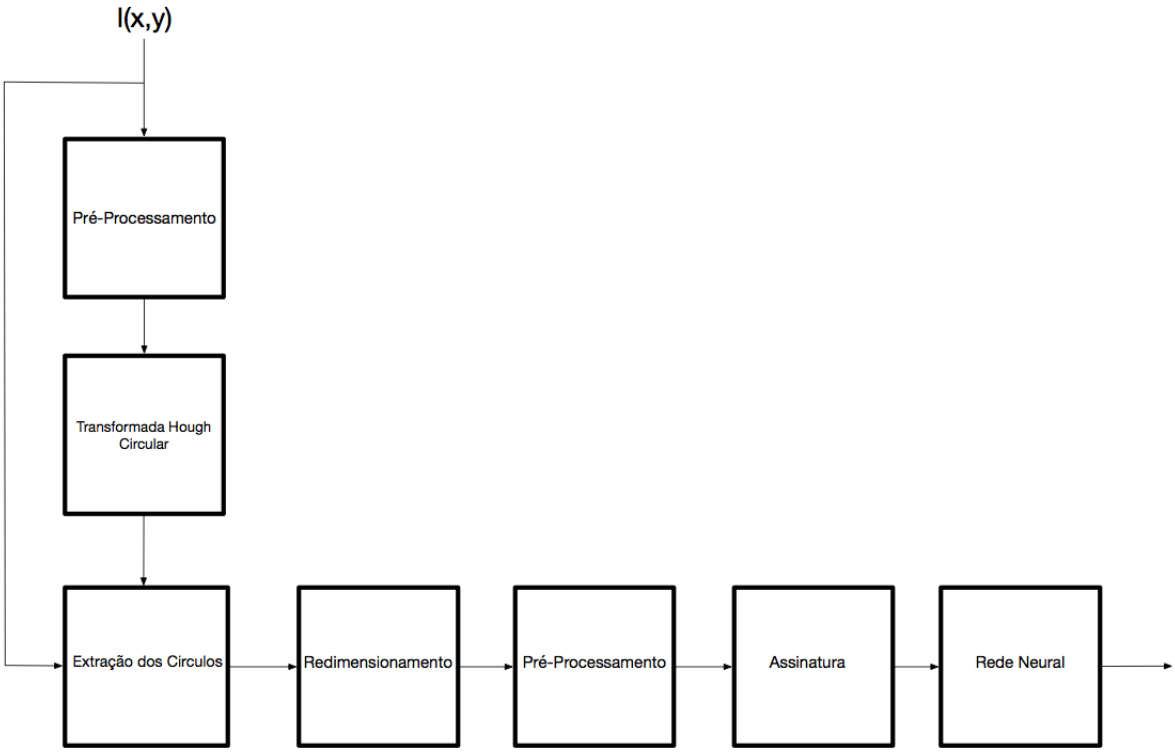


Figura 8 – Diagrama do Sistema Implementado

A metodologia utilizada no trabalho foi:

- Revisão Bibliográfica em Processamento Digital de Imagens e Reconhecimento de Padrões;
- Proposição do sistema;
- Aquisição de imagens para banco de dados e teste;
- Aquisição e Análise dos Resultados;
- Preparação do Relatório.

A implementação do sistema foi totalmente feita no MATLAB® e no seu Toolbox de Processamento de Imagens.

As imagens para o banco de dados foram tiradas na Depet Reciclagem que situa-se na Avenida Dilson Funaro S/N em Queimadas - PB. Foram tiradas 64 fotos, dessas fotos, foram gerados 217 círculos para o banco de dados. A quantidade de parafusos é de 106 e de não-parafusos é de 111. Os parafusos a serem detectados são do tipo estrela.

As fotos foram obtidas através da câmera do smartphone HTC One M8 e foram convertidas em preto e branco antes do pré-processamento.

Acerca das imagens do círculos, estas foram redimensionadas para 128x128, conforme a seção 2.3 para a assinatura. Foi utilizado a interporlação bicúcbica para o redimensionamento.

Apesar da quantidade de imagens ser relativamente pequeno, vale salientar que para essa aplicação específica, a classificação não é para detectar um parafuso genericamente, mas sim de fazer a detecção em meio à círculos detectados em imagens similares às utilizadas no treinamento da rede, em um contexto de indústria.

No que diz respeito na forma que os dados foram organizados, conforme o que está escrito no script 6.4, todas as imagens de bordas dos círculos foram salvas em png e nomeadas da seguinte maneira: se a imagem representasse um parafuso, o seu nome começaria com a letra P, caso contrário, N, após depois disso foi colocado um número, referenciando-as.

Acerca das assinaturas, conforme o que está descrito no script 6.5, as assinaturas de todas as imagens são obtidas e guardadas em uma matriz. Cada coluna da matriz representa a assinatura de uma imagem, que é um vetor de 12 posições – equivalente ao número de linhas da matriz.

A rede neural perceptron multicamada utilizada contém 12 entradas para vetores de tamanho 12, 36 percéptrons na camada escondida e dois perceptrons na camada de saída, cuja função é identificar se a assinatura na entrada é ou não é de um parafuso. A formulação dos dados alvo no treinamento supervisionado é: se a assinatura na entrada for de um parafuso, o vetor que representa o valor verdadeiro e esperado na saída é $[1 \quad 0]^T$, caso contrário, $[0 \quad 1]^T$. No treinamento, foram utilizados 15% dos dados como validação e 35% dos dados para teste e os 50% restantes para treino.

Os dados alvo são uma concatenação dos vetores descritos anteriormente onde o número de colunas é o mesmo dos dados de assinatura. Cada índice de coluna na matriz alvo corresponde à saída desejada na matriz no vetor de mesma coluna na matriz de assinaturas.

4 Resultados e Discussões

4.1 Resultado

O resultado a ser mostrado será a tabela de confusão para os dados de teste, que foi obtida após o treinamento da rede neural. Os resultados estatísticos para 40 processos de treino estão na tabela 1 e resultados subsequentes estão na tabela 2. O conjunto de treino, validação e teste foram selecionados aleatóriamente a cada processo. O formato apresentado em cada elemento é: média (desvio padrão) e na linha abaixo a quantidade média absoluta de círculos utilizados nos dados de testes. O termo positivo se referem a classe parafusos.

Tabela 1 – Tabela de Confusão de Teste

		Classe Alvo		
		Parafuso	Não-Parafuso	Total
Classe de Saída	Parafuso	36.87% (4.63%) 80.0	18.94% (5.40%) 41.1	55.81% 121.1
	Não-Parafuso	11.34% (5.35%) 24.6	32.85%(5.03%) 71.3	44.19% 95.9
	Total	48.21% 104.6	51.79% 112.4	100% 217

Tabela 2 – Resultados Complementares

Taxa de Positivos Verdadeiros	76.48%
Taxa de Negativos Verdadeiros	63.43%
Valor Preditivo Positivo	66.06%
Valor Preditivo Negativo	74.35%
Precisão	69.72%

4.2 Parâmetros e Resultados

No processo de obtenção de resultados, alguns parâmetros foram variados e a suas respectivas matrizes de confusão obtidas e analisadas, que estão listadas abaixo:

- Um número maior de perceptrons na camada escondida não acarreta na melhora dos resultados;
- O aumento do tamanho do vetor de entrada – que é dado pela diminuição do intervalo de discretização de θ na assinatura – também não acarreta na melhora dos resultados, até chegar um ponto em que começa a haver uma piora.

4.3 Discussões

No sistema, alguns parafusos não foram detectados pela Transformada Hough Circular e o motivo foi a falta de contraste entre parafuso e o fundo para a detecção das bordas. Os dois exemplo principais estão nas figuras 9 e 10. No primeiro caso, percebe-se que a cor dos parafusos é a mesma do fundo, no segundo, apesar da cor da madeira e do parafuso serem diferentes, a tonalidade dos dois quando mapeados de RGB para escala de cinza tornam-se parecidas.

Figura 9 – Exemplo - Parafuso Não-detectado

Figura 10 – Exemplo - Parafuso Não-detectado

Outro problema encontrado foi a baixa taxa de acerto, como mostrado na tabela de confusão para os dados de teste. Isso está diretamente relacionado com o processo de descrição da imagem. Da maneira que foi pensada, o processo de descrição supunha inicialmente que todos os parafusos estrela, por terem o o formato de ranhuras parecidas, teriam também uma assinatura parecida umas com as outras e que fossem algo similar com o mostrado na figura 4. Porém, na grande maioria dos parafusos, há bordas internas na ranhuras que são fortes o suficiente para serem detectadas pelo detector de bordas, algumas são até mais fortes que algumas regiões de bordas que separam a parte externa do parafuso das suas ranhuras.

As bordas encontradas dentro das ranhuras são devido a: ferrugem, alto-relevo colocado pelo próprio fabricante, sujeira, ruído que não foi retirado e algumas regiões que apresentam uma cor ligeiramente diferente por conta do impacto maior com chave de fendas.

Alguns exemplos de bordas de parafusos e suas respectivas assinaturas encontram-se nas figuras 11, 12, 13 e 14. As figuras 11 e 12 representam as bordas de parafuso de um parafuso sem muitas bordas dentro de suas ranhuras e as imagens 13 e 14 representam o oposto.

Não houve banco de dados que disponível na internet pudesse ser utilizado como dados para o sistema.

No trabalho, foi separado um conjunto de dados de parafusos que apresentavam poucas bordas dentro de suas ranhuras, que serviram de dados de treinamento juntamente com os círculos que não são parafusos para uma rede, mas os resultados da tabela de confusão não foram consistentes por falta de uma quantidade relevante de dados.

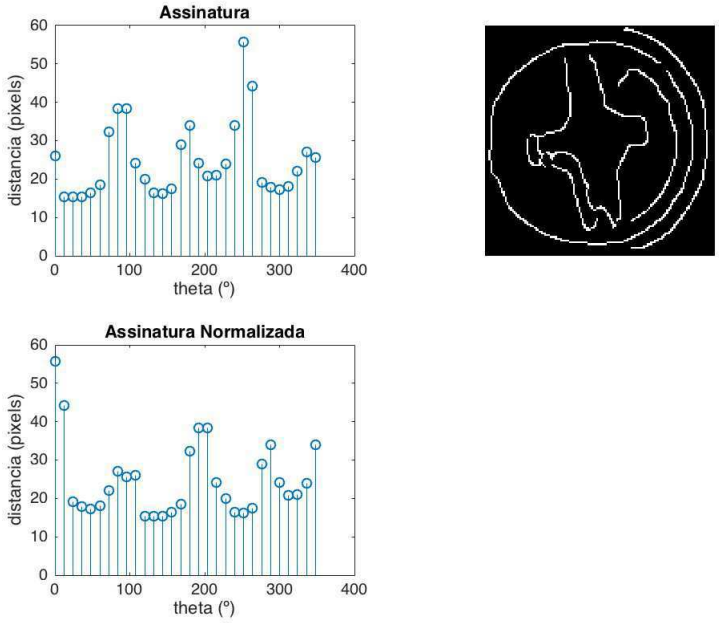


Figura 11 – Bordas de um Parafuso e sua Assinatura - Desejado

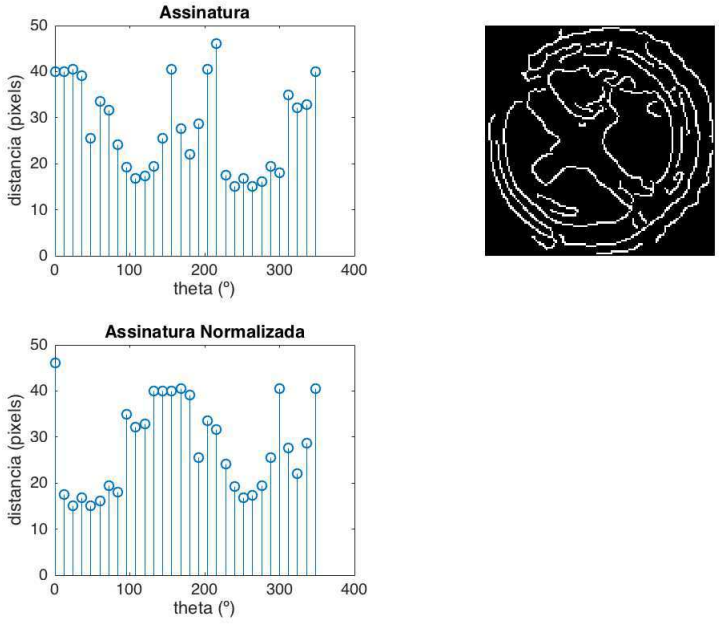


Figura 12 – Bordas de um Parafuso e sua Assinatura - Desejado

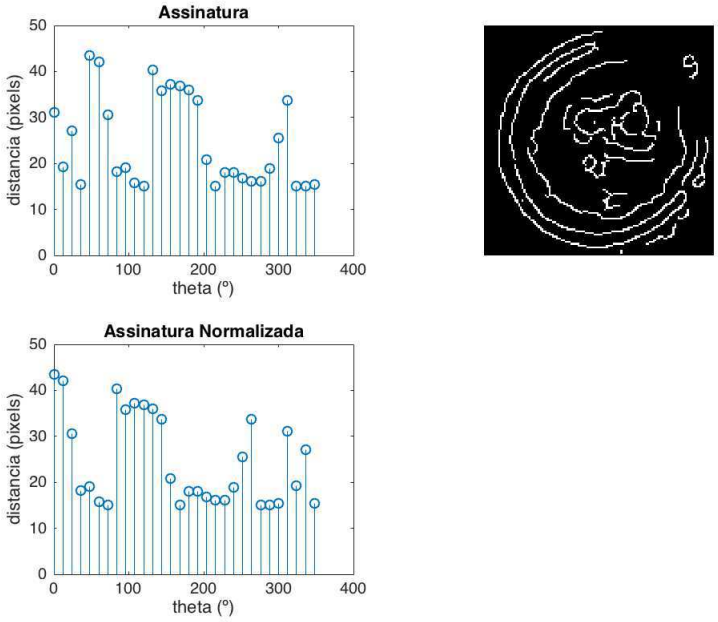


Figura 13 – Bordas de um Parafuso e sua Assinatura - Não Desejado

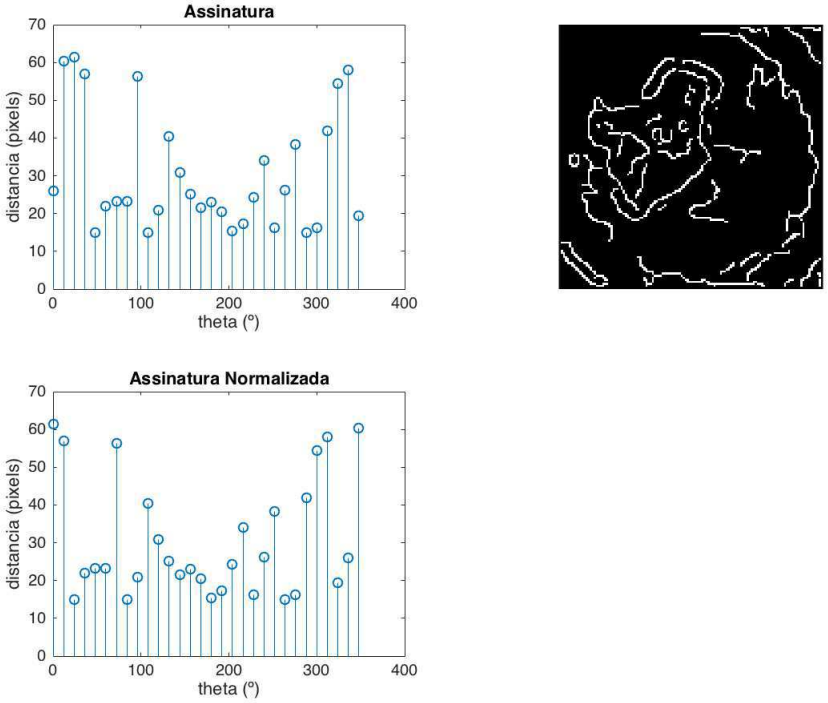


Figura 14 – Bordas de um Parafuso e sua Assinatura - Não Desejado

5 Conclusão e Sugestões para Trabalhos Futuros

Neste trabalho foi proposto, implementado e testado um sistema de processamento de imagens para detecção de parafusos. Os dados utilizados para teste foram obtidos em um ambiente típico de indústrial.

Os resultados mostraram que, mesmo para um ambiente onde não há controle dos parafusos utilizados, ou seja, o conjunto de dados de parafusos não apresentam necessariamente um processo de descrição parecidos, houve uma taxa razoável de acertos nos dados de teste, o que é interessante para um primeiro trabalho.

Para um trabalho futuro nesse mesmo tema, pode-se testar o sistema para uma quantidade maior de dados, para uma verificação mais sólida deste, bem como testá-lo para uma quantidade mais controlada de dados, ou seja, dados que representem mais parafusos parecidos entre si e/ou não tenham bordas dentro de suas ranhuras, como discutido na seção 4.3. Pode-se ainda adicionar mais classes na rede neural para outros tipos de parafusos.

A realização deste trabalho possibilitou o emprego de diversos conhecimentos aprendidos na graduação de Engenharia Elétrica alguns diretamente, como Processamento Digital de Sinais e Programação, e outros indiretamente, que serviram como base na revisão bibliográfica no que diz respeito a tópicos não vistos na graduação, como alguns tópicos de Processamento Digital de Imagens e Aprendizagem de Máquina.

Referências

1 DAVIES, E. *Machine Vision: Theory, Algorithms, Practicalities*. [S.l.]: Elsevier Inc, 1990.

2 SCREWS | Screws, Nails and Fixings | screwfix.com. 2017. Acessado em 16 de Agosto de 2017. Disponível em: <http://s7g3.scene7.com/is/image/ae235/cat840050_1?$catImages$>.

3 GONZALEZ, R. C.; WOODS, R. E. *Digital Image Processing*. [S.l.]: Pearson Education, 2008.

4 10 misconceptions about Neural Networks. 2014. Acessado em 16 de Agosto de 2017. Disponível em: <http://www.turingfinance.com/misconceptions-about-neural-networks/>.

5 PARKER, J. R. *Algorithms for Image Processing and Computer Vision*. [S.l.]: Wiley Computer Publisher, 1997.

6 ATHERTON, T. J. Size invariant circle detection. *Image and Vision Computing*, 1998.

7 BISHOP, C. M. *Neural Networks for Pattern Recognition*. [S.l.]: Oxford: Oxford University Press, 1995.

6 Anexos

6.1 Função - Assinatura

```
function [theta d] = signature(img, res_theta, res_d, d_min);

%Calcula a assinatura de uma imagem 128x128
 theta = 2*pi*[0:res_theta-1]/res_theta;
 d = zeros(size(theta));

 for i = 0: res_theta-1



     j = ceil(res_d*d_min);
      % o j computa uma distancia minima inicial, objetivo: desprezar bordas
      %que estao muito perto do centro e que
      %nao fazem parte das bordas entre ranhuras e
      %parte externa.
     while (  img( 64 + ceil(  (j/res_d)*sin(2*pi*i/res_theta)   ), 64 + ceil(   (j/res_d)*cos(2*pi
         *i/res_theta)   )  ) ~= 1 )


         j = j+1;

         if( 64 + ceil(  (j/res_d)*sin(2*pi*i/res_theta)) > 127 ||  64 + ceil(   (j/res_d)*cos(2*pi
             *i/res_theta)   )  > 127   ||...
             64 + ceil(  (j/res_d)*sin(2*pi*i/res_theta)) < 1  ||  64 + ceil(   (j/res_d)*cos(2*pi*
                 i/res_theta)   )  < 1     )
             break;
         end

     end

     d(i+1) = (j/res_d);

 end
```

6.2 Função - Filtro Gaussiano

```
function filt_img = gaussian_filter(img, sigma2)
% Filtra uma imagem com um filtro passa baixo gaussiano com parmetro sigma2
IMG = fft2(img);

size_img = size(IMG);

[x y] =  meshgrid(0 : size_img(2) - 1, 0 : size_img(1) - 1);

gauss_filter = exp(-2*pi*pi*sigma2*((x/size_img(2) - 1/2).^2   +  (y/size_img(1) - 1/2).^2  ));
```

```
gauss_filter = fftshift(gauss_filter);

filt_img = real(ifft2(gauss_filter.*IMG));
```

6.3 Função - Normalização - Assinatura

```
function norm = max_norm(in)
%aplica um deslocamento circular, mapeando o maximo para o indice 1.

[M N] = size(in);

if(M == 1)
    in = in';
end

[dummy I] = max(in);
norm =  circshift(in,-I+1);

if(M == 1)
    norm = norm';
end
```

6.4 Círculos de Fotos para Banco de Dados

```
%% Script para Extrair os Circulos das imagens de Banco de Dados
clear;
clear all
clc
close all

res_circles = 128;
scale = 40;
screw_scale = 4;

%Ha duas variaveis que contabilizam os circulos que sao ou nao parafusos
index_TF = load('index_data.mat');

T = index_TF.T;
F = index_TF.F;

for i = 1:64;

filename = [num2str(i), '.jpg']
inpic = imread(filename);
inpic = rgb2gray(inpic);

I = gaussian_filter(inpic,scale); %FPB, suavizar, melhorar SNR

[I_edges] = edge(I,'canny'); %Detector de Canny
```

```
figure(1)
subplot(2,1,1)
imshow(inpic);


subplot(2,1,2)
imshow(I_edges);

pause;
close;

[centers1, radii1, metric1] = imfindcircles(I_edges,[17 50]);%Transf. Hough
[centers2, radii2, metric2] = imfindcircles(I_edges,[51 100]);
%[centers3, radii3, metric3] = imfindcircles(I_edges,[12 24]);
[centers4, radii4, metric4] = imfindcircles(I_edges,[100 150]);

centers = [ centers1; centers2; centers4]
radii   = [ radii1; radii2; radii4]
metric  = [ metric1; metric2; metric4]


RGB = insertShape(inpic,'circle',[centers, radii]);
imshow(RGB)

pause;
close;
%% Ideia: extrair os circulos e coloca-los numa matriz 128x128
C_Edges             = zeros(res_circles,res_circles,length(radii));
Circles_Initial_Img = zeros(res_circles,res_circles,length(radii));

size_I = size(inpic);
empty_I = [];

for i = 1: length(radii)


    x1 = ceil(centers(i,2) - radii(i));
    x2 = ceil(centers(i,2) + radii(i));


    y1 = ceil(centers(i,1) - radii(i));
    y2 = ceil(centers(i,1) + radii(i));
    %Os ifs servem para continuar o loop sem fazer as atribuicoes caso as
    %dimensoes que vao ser atribuidas ao circulos estejam fora da resolucao
    %da imagem
    if(x1 < 1)

        continue;
    end
    if(x2 > size_I(1))

        continue;
    end

    if(y1 < 1)

        continue;
```

```
    end

    if(y2 > size_I(2))

        continue;
    end

    af = inpic (x1:x2, y1:y2);
    Circles_Initial_Img(:,:,i) = imresize(af, [res_circles res_circles]);

    C_filt = gaussian_filter(Circles_Initial_Img(:,:,i),screw_scale); %FPB, suavizar, melhorar SNR
    [C_edge] = edge(C_filt,'canny');

    C_Edges(:,:,i) = C_edge;


end
empty_I

dim3_L = length(radii) - length(empty_I);
C_Edges = C_Edges(:,:,1:dim3_L);
Circles_Initial_Img = Circles_Initial_Img(:,:,1:dim3_L);

%% Classificador Manual
  for i = 1:dim3_L

      figure(1)
      subplot(1,2,1)
      imshow(double(C_Edges(:,:,i)));
      set(gcf, 'Position', get(0,'Screensize'));

         subplot(1,2,2)
            imshow(Circles_Initial_Img(:,:,i),[min(min(min(Circles_Initial_Img))), max(max(max(
                Circles_Initial_Img)))], 'InitialMagnification', 'fit')
      set(gcf, 'Position', get(0,'Screensize'));


      prompt = 'A imagem e ou n e parafuso? [P/N]?';
      x = input(prompt, 's');

      if(x == 'P')
          filename = [x,num2str(index_TF.T), '.png'];
          index_TF.T = index_TF.T + 1

      elseif(x == 'N')
          filename = [x,num2str(index_TF.F), '.png'];
          index_TF.F = index_TF.F + 1

      elseif(x== 'C')
          continue;
      end
      %salvar a imagem de borda com um nome que faca referncia   classificacao
      imwrite(C_Edges(:,:,i),filename)
  end

  T = index_TF.T;
  F = index_TF.F;
```

```
end

  save index_data.mat T F
```

6.5 Banco de Dados para Assinatura

```
%% Script para Obtencao e Armazenamento das Assinaturas a partir do Banco de Circulos
clear all
close all
clc

theta_res = 30;
p_res = 15;
d_min = 15;

index_TF = load('index_data.mat');


data_sigs = [];
%% Extrair assinatura de parafusos
for i = 1:index_TF.T -1

    filename = ['P' , num2str(i), '.png'];
    A = imread(filename);

    A = (A ~= 0);

    [theta, sig] = signature(A, theta_res,p_res, d_min);
%% plotagem das bordas, assinatura e assinatura normalizada
%     figure(i)
%     subplot(2,2,1)
%     stem(180*theta/pi, sig);
%     xlabel('theta ( )');
%     ylabel('distancia (pixels)')
%     title('Assinatura')
%
     sig = max_norm(sig);
%
%     subplot(2,2,3)
%     stem(180*theta/pi, sig);
%     xlabel('theta ( )');
%     ylabel('distancia (pixels)')
%     title('Assinatura Normalizada')
%
%     subplot(2,2,2)
%     imshow(A)
%
%     pause;
%     close;

    sig = sig';

    assinatura = ['S' , num2str(i), '.mat' ]   ;
```

```
    data_sigs = [data_sigs, sig]    ;
end

%extrair assinatura de n-parafusos.
for i = 1:index_TF.F -1

    filename = ['N' , num2str(i), '.png'];
    A = imread(filename);

    A = (A ~= 0);

    [theta, sig] = signature(A, theta_res,p_res, d_min);

    sig = max_norm(sig);

%     figure(i)
%     subplot(1,2,1)
%     plot(180*theta/pi, sig);
%
%     subplot(1,2,2)
%     imshow(A)
%
%     pause;
%     close;

    sig = sig';

    assinatura = ['S' , num2str(i), '.mat' ]   ;

    data_sigs = [data_sigs, sig]   ;
end
    save('Assinaturas','data_sigs');

    %definindo o vetor de classifica para treino
    des_out = [ones(1,index_TF.T - 1) zeros(1,index_TF.F - 1)];
    des_out = [des_out; ~des_out];

    save des_out.mat des_out;
```

6.6 Visualização de Pré-Processamento

```
%% Script para visualizar e analisar como o sistema atua em cada imagem
clear;
clear all;
clc;
close all;
%% parametros
res_circles = 128; %resolucao para os circulos normalizados
scale = 32; %sigma2 - filtro gaussiano para imagem
scale_screw = 4; %sigma2 - filtro-gaussiano - processamento do parafuso
T = 0; %limiar para deteccao de bordas
index_TF = load('index_data.mat');
```

```
inpic = imread('2.jpg');
inpic = rgb2gray(inpic);

%inpic = medfilt2(inpic);
%% pre-processamento
I = gaussian_filter(inpic,scale); % filtragem

[I_edges] = edge(I,'canny'); %deteccao de bordas

figure(1)
subplot(2,1,1)
imshow(inpic);


subplot(2,1,2)
imshow(I_edges);

pause;
close;
%% Transformada Hough Circular e Deteccao de Circulos
[centers1, radii1, metric1] = imfindcircles(I_edges,[12 50]);%Transf. Hough
[centers2, radii2, metric2] = imfindcircles(I_edges,[51 100]);
%[centers3, radii3, metric3] = imfindcircles(I_edges,[12 24]);
[centers4, radii4, metric4] = imfindcircles(I_edges,[100 150]);

centers = [ centers1; centers2; centers4]
radii = [ radii1; radii2; radii4]
metric = [ metric1; metric2; metric4]


RGB = insertShape(inpic,'circle',[centers, radii]);
imshow(RGB)

pause;
close;

%% Normalizacao para conjunto de matrizes 128x128
Circles  = zeros(res_circles,res_circles,length(radii));
Screws = zeros(res_circles,res_circles,length(radii));

empty_I = [];

size_I = size(inpic);

for i = 1: length(radii)

    x1 = ceil(centers(i,2) - radii(i));
    x2 = ceil(centers(i,2) + radii(i));
    y1 = ceil(centers(i,1) - radii(i));
    y2 = ceil(centers(i,1) + radii(i));

    if(ceil(centers(i,2) - radii(i)) < 1)
        empty_I = [i;empty_I];
        continue;
    end
    if(ceil(centers(i,2) + radii(i)) > size_I(2))
```

```
        empty_I = [i;empty_I];
        continue;
    end

    if(ceil(centers(i,1) - radii(i)) < 1)
        empty_I = [i;empty_I];
        continue;
    end

    if(ceil(centers(i,1) - radii(i)) > size_I(1))
        empty_I = [i;empty_I];
        continue;
    end

    af = I_edges(x1 : x2 , y1:y2);
    Circles(:,:,i) = imresize(af, [res_circles res_circles]);

    Screw =  inpic(x1 : x2 ,y1:y2);

    Screws(:,:,i) = imresize(Screw, [res_circles res_circles]);
end

  %% Mostrar resultados relevantes do que foi tratado - parafusos e bordas
  for i = 1:length(radii)

      if(sum(empty_I == i))
          continue;
      end

      %Pre processamento
      Screw_filt = gaussian_filter(Screws(:,:,i),scale_screw);
      Screw_Edge = edge(Screw_filt,'canny');
      %obtencao de resultados
      figure(i)
      subplot(2,2,1)
     fit_and_display(double(Screw_Edge));
     subplot(2,2,2)
      fit_and_display(Screws(:,:,i));
   %   set(gcf, 'Position', get(0,'Screensize'));

%       figure(2)
%            imshow(Circles_Initial_Img(:,:,i),[min(min(min(Circles_Initial_Img))), ...
%max(max(max(Circles_Initial_Img)))], 'InitialMagnification', 'fit')
%       set(gcf, 'Position', get(0,'Screensize'));
%


  end
```