MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

SISTEMA DE VISÃO COMPUTACIONAL PARA IDENTIFICAÇÃO DE POSIÇÃO E ORIENTAÇÃO DE PEÇAS DISPOSTAS ALEATORIAMENTE NA ÁREA DE TRABALHO DE UM ROBÔ DE MANIPULAÇÃO

por

Leandro Tognon

Dissertação para obtenção do Título de
Mestre em Engenharia

Porto Alegre, Março de 2019

SISTEMA DE VISÃO COMPUTACIONAL PARA IDENTIFICAÇÃO DE POSIÇÃO E ORIENTAÇÃO DE PEÇAS DISPOSTAS ALEATORIAMENTE NA ÁREA DE TRABALHO DE UM ROBÔ DE MANIPULAÇÃO

por

Leandro Tognon
Engenheiro Mecânico

Dissertação submetida ao Corpo Docente do Programa de Pós-Graduação em Engenharia Mecânica, PROMEC, da Escola de Engenharia da Universidade Federal do Rio Grande do Sul, como parte dos requisitos necessários para a obtenção do Título de

Mestre em Engenharia

Área de Concentração: Processos de Fabricação

Orientador: Prof. Dr. Flávio José Lorini
Co-orientador: Prof. Dr. Renato Ventura Bayan Henrique

Aprovada por:

Prof. Dr. José Antônio E. Mazzaferro, PROMEC/UFRGS

Prof. Dr. Leandro Costa de Oliveira, DEM/UFSM

Prof. Dr. Alcy Rodolfo dos Santos Carrara, DEMEC/UFRGS

Prof. Dr. Fernando Marcelo Pereira
Coordenador do PROMEC

Porto Alegre, 29 de Março de 2019

AGRADECIMENTOS

Agradeço a minha família, meus pais e irmão, aos meus amigos e todos aqueles que de alguma forma participaram dessa conquista, me auxiliando ou simplesmente me apoiando com palavras de incentivo.

Agradeço ao orientador, Prof. Dr. Flávio José Lorini, e ao co-orientador Prof. Dr. Renato Ventura Bayan Henrique, pelo apoio e auxílio no desenvolvimento do trabalho.

Agradeço à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo suporte financeiro para a elaboração deste trabalho.

RESUMO

Nos últimos anos, a utilização de robôs na indústria vem crescendo significativamente. Cada vez mais as empresas utilizam os robôs em tarefas repetitivas que são previamente programadas, executando sempre o mesmo movimento, sem nenhuma flexibilidade. Para resolver essa falta de flexibilidade, sistemas de visão computacional tem sido desenvolvidos para trabalhar em conjunto com robôs, dando aos mesmos a capacidade de observar a área de trabalho através das lentes de uma câmera e, utilizando ferramentas de processamento de imagem, interpretar o que estão "enxergando", flexibilizando seu uso. Neste trabalho é apresentado um sistema de visão computacional capaz de, através da captura e processamento de imagens digitais, identificar a posição e orientação de vários objetos dispostos aleatoriamente no mesmo plano da área de trabalho. Primeiramente a imagem é capturada por uma câmera e enviada ao software MATLAB®, onde é realizada a limiarização dos mesmos, afim de obter-se uma imagem final com os objetos segmentados e identificados. Outros processos como dilatação morfológica das imagens e rotulagem de objetos são utilizados nessa etapa. A partir da imagem segmentada, utilizando uma função do MATLAB® chamada *regionprops*, são encontradas as posições dos centros de massa de cada objeto, e utilizando uma rede neural artificial determina-se a orientação dos mesmos. A rede neural utilizada possui arquitetura multi-camada, com uma camada escondida e alimentação para frente (*feed forward*), treinada previamente com várias imagens de peças semelhantes às que seriam utilizadas para os testes. As coordenadas obtidas são enviadas ao controlador de um manipulador robótico, que realiza a coleta dessas peças e as coloca em outro local previamente definido. Esse sistema de visão acrescenta ao robô a capacidade de coletar peças dispostas aleatoriamente em um plano, sem a necessidade prévia de informar suas coordenadas. O método proposto foi testado utilizando algumas imagens que simulam uma situação real em um ambiente industrial, com várias peças dispostas de forma aleatória em um plano, onde foram comparados os valores para a posição e orientação encontrados pelo método com os valores reais. Também foi avaliado o tempo de processamento total necessário para obter as coordenadas e movimentar as peças. Os testes mostraram que o método apresenta resultados adequados e robustez nas respostas para diferentes cenários.

Palavras-chave: Processamento de imagens; Redes neurais artificiais; Visão computacional, Robôs industriais.

ABSTRACT

In recent years, the use of robots in the industry has been growing significantly. More and more companies use the robots in repetitive tasks that are pre-programmed, always executing the same routines, without any flexibility. To fix this lack of flexibility, computer vision systems have been developed to work in conjunction with robots, giving them the ability to observe the work area through a camera lens, and using image processing tools interpret what they are "seeing", making their use more flexible. This work presents a computer vision system capable of capturing and processing digital images, identifying the position and orientation of several objects, randomly arranged in the same level of the work area. First, the image is captured by a camera and sent to MATLAB® software, where the image is thresholded, in order to obtain a final image with the objects segmented and identified. From the segmented image, using a MATLAB® function called regionprops, we find the position of the centre of mass of each object, and using an artificial neural network, we determine their orientation. The neural network used has a feed forward multi-layer architecture, with one hidden layer, that was previously trained with several images of pieces, similar to those used for the tests. The coordinates obtained are sent to a robotic manipulator controller, which performs the collection of these objects and places them in another pre-defined place. This vision system give to the robot the ability to collect randomly arranged pieces in the same surface, without the prior need to inform its coordinates. The proposed method was tested using some images that simulate a real situation in an industrial environment, with several pieces randomly arranged in the same surface, where the values found by the method for the position and orientation of the pieces were compared with the real values. The total processing time required to obtain the coordinates and to move parts was also evaluated. The tests showed that the method shows good results and robustness with different scenarios.

Keywords: Image processing, Artificial neural network; Computer vision, Industrial robots.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

A/D	*Analog-digital*
ADC	*Analog-digital converter*
ANN	*Artificial Neural Network*
CCD	*Charge-coupled Device*
CHF	*Circular harmonic function*
CMOS	*Complementary Metal-oxide Semiconductor*
CNN	*Convolutional Neural Network*
DOF	*Degree of freedom*
FCN	*Fully convolutional network*
GDL	Graus de liberdade
GMM	*Gaussian mixture model*
GPM	*Geometric pattern matching*
HD	*High Definition*
IFR	*International Federation of Robotics*
IP	*Internet Protocol*
ISO	*International Organization for Standardization*
LAR	Laboratório de Automação e Robótica
MATLAB®	*Matrix Laboratory*
PLC	*Point Cloud Library*
PROMEC	Programa de Pós-Graduação em Engenharia Mecânica
R-CNN	*Region-based Convolutional Neural Network*
RAM	*Random Access Memory*
RBP	*Random bin-picking*
RGB	*Red, Green, Blue*
RNA	Rede Neural Artificial
RPN	*Region proposal network*
SCARA	*Selective Compliance Assembly Robot Arm*
SIFT	*Scale Invariant Feature Transform*
SVM	*Support vector machine*

TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
UFRGS	Universidade Federal do Rio Grande do Sul
USB	*Universal Serial Bus*
YOLO	*You only look once*

LISTA DE SIMBOLOS

a	Aceleração do manipulador
c	Posição da coluna na imagem digital
f	Distância focal da câmera
F	Ponto focal
$f(x,y)$	Função que representa a imagem digital
$g(x,y)$	Função que representa a imagem após operação morfológica
L	Nível de cinza de um ponto na imagem digital
r	Posição da linha na imagem digital
r_p	Relação pixels/mm
SE	Elemento estruturante
T	Limiar da escala de cinza nas imagens digitais
U	Ângulo de rotação do efetuador do manipulador
v	Velocidade do manipulador
w_n	Pesos das entradas dos neurônios
x	Excentricidade entre o eixo focal da câmera e o efetuador na direção X
X	Posição x no sistema de referência do manipulador
x_n	Valor de entrada do neurônio
y	Excentricidade entre o eixo focal da câmera e o efetuador na direção Y
Y	Posição y no sistema de referência do manipulador
y_n	Valor de saída do neurônio
Z	Posição z no sistema de referência do manipulador
θ	Limiar da função do neurônio
$\emptyset(\cdot)$	Função de ativação do neurônio

1 INTRODUÇÃO

Desde o final do século 18 a indústria vem passando por grandes mudanças nos seus sistemas de produção. Primeiro foi a introdução de máquinas a vapor e hidráulicas, durante a primeira Revolução Industrial. No início do século 20 foi introduzida a produção em massa baseada na divisão de trabalho e utilização da energia elétrica, que deu origem a segunda Revolução Industrial. Na Terceira Revolução Industrial, no início dos anos 1970, foram introduzidos o uso de eletrônicos e da tecnologia da informação para automação da produção. Atualmente o mundo vem passando pela quarta revolução industrial, também chamada de Indústria 4.0, que é baseada na aplicação de sistemas ciber-físicos, sistemas onde há uma profunda integração entre robótica, inteligência artificial, automação, realidade aumentada, nanotecnologia, transferência de dados, internet das coisas e tecnologias de fabricação.

A utilização de robôs em atividades que antes eram realizadas por seres humanos está cada vez mais ampla, e futuramente, cada vez mais tarefas serão realizadas por robôs autônomos, que necessitam perceber o que está acontecendo ao redor, para que possam realizar as tarefas de forma satisfatória e segura. Conforme aumenta a complexidade da tarefa, mais recursos os sistemas robóticos devem possuir, sejam eles sensores, sistemas de visão, módulos de controle avançado, etc.

Nos seres humanos, o sentido da visão é importante para perceber o que está acontecendo no mundo ao redor, entender a situação em que se encontra e auxiliar na tomada de decisões. Nos robôs, esse sentido é agregado através da utilização de algumas técnicas de visão computacional, tais como visão estéreo, fotogrametria, luz estruturada, triangulação laser, entre outras, através do processamento de imagens adquiridas, e conversão das informações obtidas para linguagem do robô. Essa capacidade de o robô “enxergar” possibilita o movimento no volume de trabalho evitando obstáculos, possibilita trabalhar em modo colaborativo com seres humanos e outros robôs, identificar e localizar objetos, etc.

Dentro da área da visão computacional, a segmentação de imagens é o processo mais crítico, pois o sentido de separar e identificar objetos, que para os humanos é simples e natural, para as máquinas é interpretativo e baseado nos dados obtidos por meio da captura das imagens. A segmentação de imagens é utilizada em diversas áreas, desde áreas relacionadas a medicina e saúde, como contagem de células, identificação de tipos de células e tecidos, identificação de

tumores, etc., até áreas relacionadas a indústria, como em identificação de peças em processos industriais, inspeção e avaliação da qualidade de peças, manipulação de partes em processos de fabricação, entre outras atividades.

Atualmente, devido a flexibilidade exigida dentro das indústrias, a utilização de robôs para desenvolver algumas tarefas tornou-se de extrema importância para o desenvolvimento de sistemas de fabricação mais ágeis e confiáveis. Nesse contexto, sistemas que utilizam visão computacional, tais como sistemas autônomos de *pick and place* (pega e põe) e/ou *bin picking* (pega na cesta), estão sendo desenvolvidos para realizar diversas tarefas, onde se destacam as operações de montagem, empacotamento, inspeção ou simplesmente pegar a peça em um local e colocá-la em outro.

O trabalho proposto utiliza um sistema de visão computacional para identificar imagens de peças semelhantes em um campo de visão predefinido, processando as imagens capturadas e identificando a posição e orientação de cada uma das peças. Posteriormente essas coordenadas são enviadas para o controlador de um robô SCARA, que então deve coletar cada uma das peças e as transportar para um local determinado, com a posição e orientação predefinida, caracterizando um processo de *bin picking*.

Segundo Sansoni et al., 2014, os principais desafios para um sistema de *bin picking* efetivo são: capturar a cena através do sistema de visão, segmentar essa cena, separando os diferentes objetos, e reconhecer e estimar as poses dos objetos reconhecidos.

1.1 Descrição do problema

Sistemas para reconhecimento de objetos estão sendo amplamente estudados e desenvolvidos para obter informações precisas sobre a posição e orientação de peças no ambiente industrial. Essas informações são utilizadas juntamente com sistemas automatizados para realização de tarefas monótonas ou repetitivas, para manipulação de peças em ambientes perigosos, para movimentação de peças grandes, pesadas ou cortantes, entre outras atividades, para que não seja necessário a presença humana na execução dessas tarefas, ou simplesmente para ganho de produtividade.

Manipular peças em sistemas de montagem, classificação, inspeção, soldagem, etc., através da utilização de robôs, que possam identificar o cenário e tomar a melhor decisão para executar tal tarefa, é uma necessidade real e atual. Com a crescente necessidade de

automatização em várias atividades industriais, sistemas de visão para robôs tornaram-se uma necessidade, para que a integração homem-robô seja a mais segura possível e que os robôs se tornem cada vez mais autônomos, capazes de “pensar” e agir sem a intervenção humana.

1.2 Objetivo do trabalho

O objetivo deste trabalho é desenvolver um sistema de visão robótico para um manipulador SCARA, capaz de identificar peças com orientação e posição qualquer em uma superfície plana, e assim obter as coordenadas de posição e orientação das mesmas, enviando as informações para o controlador do robô, que por sua vez coleta as peças e posiciona-as de acordo com a tarefa estipulada. O sistema tem as seguintes características:

- utilização de 1 câmera simples;
- captura e processamento das imagens, utilizando técnicas conhecidas;
- detecção das peças que estão dentro do espaço visual do sistema;
- determinação da posição e orientação das peças;
- utilização de peças de geometria similares, para detecção e manipulação;
- utilização de um software de programação de baixa complexidade;
- utilização de um manipulador SCARA;
- integração entre o sistema de visão e o controlador do robô;
- minimizar a intervenção humana;
- utilizar imagens estáticas, sem movimento relativo entre a peça e o sistema de visão.

1.3 Revisão bibliográfica

Sistemas que executam as tarefas de *pick and place* e *bin picking* podem envolver várias áreas do conhecimento, tais como captura e processamento de imagens digitais, programação computacional, reconhecimento de objetos, aprendizado de máquina, redes neurais, sistemas robóticos industriais, protocolos de comunicação, entre outras. Existem vários trabalhos e pesquisas envolvendo essas áreas de conhecimento, seja de forma específica e voltada para uma delas unicamente, seja para sistemas que agrupam mais de uma área.

Segundo Golnabi et al., 2007, com a introdução da automação nas industrias, tarefas complexas foram transformadas em simples instruções passo-a-passo que podem ser repetidas

por uma máquina. Tarefas de montagem e inspeção, por exemplo, são realizadas em diferentes processos de manufatura, e geralmente desempenhadas por trabalhadores, o que torna os sistemas de visão para máquinas mais atrativo para realização automatizada. A expectativa é que os sistemas de visão possam realizar as tarefas de adquirir a imagem e analisá-las, reconhecer algumas características ou objetos contidos na cena e explorar e impor restrições de acordo com o ambiente.

Para Consultant, 2014, o maior problema encontrado em processos automatizados de montagem é localizar e pegar uma peça na orientação correta, de um contentor no qual as peças estejam aleatoriamente posicionadas e sortidas. A capacidade de realizar essa tarefa é denominada *random bin picking (RBP)* (pegar peças aleatoriamente em um contentor). Um sistema robótico de RBP deve criar uma imagem dos objetos e do recipiente que os contém, isolar o objeto do seu plano de fundo, calcular a pose do objeto com relação ao robô ou ao sensor de captura da imagem, gerar a trajetória que leva o atuador até o objeto, e finalmente pegar o objeto e transferi-lo para o local desejado. Diversas tecnologias estão envolvidas e precisam ser integradas em um sistema robótico de RBP, entre elas está a aquisição da imagem, técnicas de iluminação, processamento de dados, manuseio das peças e instruções para o robô. As duas tecnologias mais críticas são a visão 3D e os algoritmos para interpretar as imagens adquiridas.

Nasr-Isfahani et al., 2008, propuseram um método para segmentar células que se tocam em imagens microscópicas. Para conseguir determinar o número de células presentes na imagem, a solução foi dividida em duas partes: separação das células e do plano de fundo, e separação individual das células. Para a primeira parte propõem o uso de uma técnica de segmentação baseada em grafos para separar o agrupamento de células do plano de fundo, e para complementar a segmentação é aplicado um algoritmo de limiarização baseado no método de Otsu [Otsu, 1979]. Para a segunda parte da solução, a partir da imagem binária gerada, a segmentação individual das células é obtida utilizando dois algoritmos em conjunto, um que aplica uma técnica de identificação de cantos na imagem binária, e outro que utiliza a transformada *watershed* com controle de marcadores, onde são encontradas linhas que segmentam as células individualmente.

Korath et al., 2007, propuseram um método combinado de separação de partículas que se tocam ou se sobrepõem em imagens microscópicas. Primeiramente utilizam um filtro da mediana para remoção de algum ruído que possa estar presente na imagem. Na sequência

utilizam o método de Otsu, [Otsu, 1979], para limiarizar a imagem e obter uma imagem binária, onde as partículas são separadas do plano de fundo. Para a separação das partículas que estão se tocando ou sobrepondo utilizam dois métodos em conjunto, primeiro são percorridas linhas em diversas direções na imagem original e identificados locais onde o valor de intensidade do pixel está muito abaixo em relação aos vizinhos, que é denominado de vale. Encontradas as linhas contendo os vales, é gerada uma imagem com essas linhas que é subtraída da imagem binária encontrada anteriormente, obtendo-se a separação de algumas partículas. O segundo método é a extração de características geométricas das curvas das bordas identificadas na imagem. A utilização desses dois métodos em conjunto consegue identificar as linhas que separam as partículas e segmentar satisfatoriamente a imagem.

Casasent et al., 1996, desenvolveram um algoritmo para inspeção dos grãos de pistache a partir de imagens das bandejas em uma esteira, obtidas através de raio-X. Um dos primeiros desafios era localizar individualmente cada um dos grãos, depois armazenar a imagem de cada um deles para então, utilizando um classificador, determinar a qualidade. Para realizar essa tarefa foram utilizadas técnicas de segmentação, tais como limiarização (binarização) para separar os grãos do plano de fundo, seguida por atribuição de uma cor diferente (diferentes escalas de cinza) para cada aglomerado de pixels encontrado, e através do tamanho de cada agrupamento determinar tratar-se de ruído, um grão único ou mais de um grão. Nos agrupamentos classificados como contendo mais de um grão foram aplicados filtros harmônicos de função circular (CHF) e filtro Gaussiano para determinar quantos grãos havia no agrupamento e quais eram seus centros. A partir dessas informações foi utilizado a segmentação baseada no princípio de *watershed* juntamente com a transformada da distância, e a partir do resultado foram obtidas máscaras binárias para cada um dos grãos do aglomerado, podendo separá-los individualmente para posterior classificação de qualidade.

Qin et al., 2013, propôs um método de segmentação para imagens de grãos de milho em contato, onde a imagem capturada é limiarizada (binarizada), obtendo-se a separação entre os pixels correspondentes aos grãos e os que representam o plano de fundo. A partir da imagem binária é calculada a transformada da distância, que percorre a imagem e calcula a distância entre o pixel e a borda mais próxima. A partir dessa nova imagem das distâncias aplica-se a transformada H-máxima, que limita os valores das distâncias a um limiar escolhido, e só então aplica-se a segmentação por *watershed*. A aplicação da transformada H-máxima é feita de

forma iterativa, variando os valores do limiar até obter a identificação individual dos grãos, e assim evitar a segmentação exagerada da imagem.

Scavino et al., 2007, utilizam uma técnica de segmentação baseada em algoritmo genético para resolver o problema de garrafas plásticas recicláveis sobre uma esteira e que estão se tocando, dificultando a separação dos objetos na imagem capturada. No problema assume-se que somente duas garrafas estão se tocando e que o formato das mesmas é convexo, pois assim seria possível separá-las utilizando uma linha reta. O algoritmo genético então gera linhas que unem 2 pixels das bordas das garrafas e através de um processo iterativo determina qual a linha que melhor representa a linha real que separa as duas garrafas, obtendo assim a imagem com as duas garrafas separadas (segmentadas).

Ju et al., 2018, desenvolveram um sistema de monitoramento de porcos em tempo real, para acompanhamento dos animais 24 horas por dia. Para captura da imagem foram usadas câmeras de baixo custo do modelo Microsoft Kinect que monitoravam o tempo todo a área destinada aos animais. Utilizando o modelo de mistura Gaussiana (*GMM – Gaussian mixture model*) eram identificados somente os animais que estavam se movimentando, porém, alguns animais acabavam encostando em outros, o que gerava áreas na imagem segmentada com mais de um animal. Para o desenvolvimento do trabalho foi considerado que somente 2 porcos estavam se tocando, e a partir disso a imagem segmentada desses 2 porcos se tocando era processada utilizando um sistema de detecção de objetos chamado YOLO, que utiliza uma técnica de reconhecimento de objetos baseada em redes neurais de convolução (CNN – *Convolutional Neural Network*). O módulo de processamento do YOLO encontra várias caixas de contorno com a respectiva probabilidade de existência do objeto, então o algoritmo seleciona uma ou duas dessas caixas segundo critérios de tamanho da caixa de contorno, condições das bordas dessas caixas e um critério chamado de valor de confiança. Caso as caixas de contorno sejam selecionadas, é avaliada a qualidade da segmentação e a imagem é segmentada para obter os animais individualmente. Caso não sejam alcançados os critérios anteriores, ou não sejam encontradas caixas de contorno válidas, a imagem dos animais se tocando é enviada para um módulo de processamento de imagem, onde através de morfologia matemática, geração de linhas guia e geração de pontos côncavos, a imagem é segmentada, identificando cada animal individualmente. Após a segmentação correta da imagem e identificação dos animais, são monitorados em tempo real.

Medina, 2015, desenvolveu um sistema de visão computacional aplicado a um robô cilíndrico com acionamento pneumático de 5 graus de liberdade, considerando somente uma peça localizada na área de trabalho. Após adquirida a imagem pela câmera e convertida para tons de cinza, aplica o método de Otsu para limiarização da imagem [Otsu, 1979]. A partir da imagem binária são calculados os momentos normais e centrais da peça, método proposto por Hu [Hu, 1962]. Para a determinação da posição e orientação da peça, previamente foi gerada uma tabela (*look-up table*) com o cálculo dos momentos para a mesma peça, girada a cada um grau, e através da comparação entre os valores encontrados com os valores da tabela é possível determinar o centroide da peça e a orientação da mesma, e assim obter as coordenadas a serem enviadas para o robô.

Wang et al., 2016, realizaram um estudo onde foi utilizada uma rede neural de convolução multi-nível para determinar a pose do objeto contido na imagem. O primeiro passo foi a criação de 1780 imagens sintéticas (geradas no computador) para 3 categorias de objetos, a partir das quais foi criado um banco de dados com essas imagens em diferentes pontos de vista, gerando um total de 2.670.000 imagens de treino. A partir desse banco de dados gerado foi treinada a CNN para identificar a pose do objeto em qualquer imagem real.

O reconhecimento de objetos e sua localização em imagens é um problema fundamental e um dos mais desafiadores em visão computacional. Vários métodos foram e vem sendo desenvolvidos para esse propósito.

Lowe, 2004 propôs um método para extrair características distintas e invariantes de imagens, que não sofrem alteração se a mesma rotacionar, mudar a escala ou alterar a iluminação. Essas características distintas podem ser de um objeto em particular ou uma cena, e através da comparação dessas características obtidas com um banco de dados conhecido pode-se reconhecer um objeto ou lugar contido na imagem. Esse método é conhecido como SIFT (*Scale Invariant Feature Transform*), e é capaz de detectar um objeto ou cena em imagens bastante confusas ou parcialmente encobertas.

Girshick et al., 2015, utilizam um método chamado de rede neural de convolução baseado em regiões (*R-CNN – region-based convolutional neural network*) para detectar objetos. Esse método consiste em identificar 2000 regiões da imagem utilizando um algoritmo chamado de procura seletiva [Uijlings et al., 2013], recortando e redimensionando essas regiões para o formato de dados de entrada de uma CNN. A CNN extrai as características dessas regiões em forma de vetor, posteriormente analisado em um SVM (*support vector machine*), que

determina se a região proposta é realmente um objeto, localizando-o na imagem utilizando uma caixa de contorno (*bounding box*). Ainda em 2015, Girshick propôs uma melhoria para o R-CNN, denominada de *Fast R-CNN*, onde a detecção de objetos é mais rápida. Nessa nova proposta, ao invés de identificar as 2000 regiões propostas e enviar para a CNN, a imagem toda é enviada para CNN, que gera um mapa de características e identifica as regiões a serem propostas, recortando-as e extraindo o vetor de características de cada uma delas, não sendo necessário fornecer as 2000 regiões separadamente para CNN, o que tornou o método muito mais rápido.

Ren et al., 2017, desenvolveram um método baseado no R-CNN e *Fast* R-CNN, porém sem utilizar algoritmos de procura seletiva para identificar as regiões a serem propostas. Nesse novo método é utilizada uma rede neural de proposição de regiões (RPN – *region proposal network*) integrada a *Fast* R-CNN, que recebeu o nome de *Faster R-CNN*. A RPN recebe a imagem como dado de entrada e gera um mapa de características com as regiões propostas, e a *Fast* R-CNN classifica essas regiões como sendo ou não um objeto. Esse novo método foi testado e se mostrou muito mais rápido que o R-CNN e o *Fast* R-CNN.

He et al., 2017, avançaram um passo em relação ao *Faster* R-CNN, e desenvolveram um método para, além de identificar a região que contém o objeto, gerar uma máscara delimitando somente os pixels do objeto. Esse método chamado de *Mask* R-CNN é baseado no *Faster* R-CNN, porém enquanto o Faster R-CNN gera uma caixa de contorno contendo o objeto, nesse método é utilizada uma FCN (*fully convolutional network*) em cada caixa de contorno, que identifica os pixels do objeto e gera uma máscara, possibilitando identificar quais os pixels são do objeto detectado.

2 FUNDAMENTAÇÃO TEÓRICA

Para realização do trabalho foram necessários conhecimentos sobre métodos de aquisição de imagens, imagens digitais, manipulação e programação computacional, métodos de processamento de imagens digitais, princípios de funcionamento de um robô industrial, meios de comunicação entre sistemas, redes neurais artificiais, entre outros. As principais definições e métodos clássicos e modernos, que serviram de embasamento para o desenvolvimento do trabalho, serão abordados nesse capítulo.

2.1 Imagens digitais

Segundo Marques Filho e Viera Neto, 1999, uma imagem monocromática pode ser descrita matematicamente por uma função $f(x,y)$ da intensidade luminosa, sendo seu valor, em qualquer ponto de coordenadas espaciais (x,y), proporcional ao brilho (ou nível de cinza) da imagem naquele ponto. A função $f(x,y)$ representa o produto da interação entre a iluminância $i(x,y)$, que exprime a quantidade de luz que incide sobre o objeto, e as propriedades de refletância ou de transmitância próprias do objeto, que podem ser representadas pela função $r(x,y)$, cujo valor exprime a fração de luz incidente que o objeto vai transmitir ou refletir ao ponto (x,y) (Figura 2.1).

$$f(x,y) = i(x,y).r(x,y) \qquad (2.1)$$

Onde:

$0 < i(x,y) < \infty$

$0 < r(x,y) < 1$

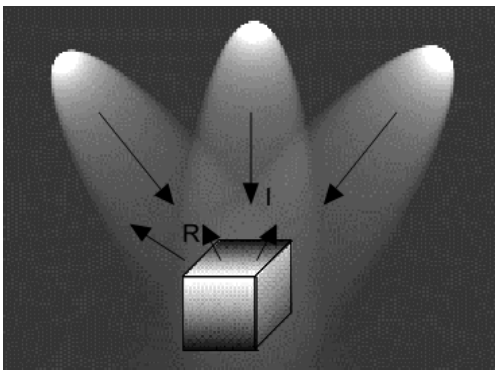


Figura 2.1 - Os componentes ilimunância (I) e refletância (R) de uma imagem

Ao longo do trabalho, a intensidade de uma imagem monocromática na coordenada (x, y) será denominada nível de cinza L da imagem naquele ponto, e estará dentro do intervalo $L_{min} \leq L \leq L_{max}$, que normalmente é representado pelo intervalo de valores inteiros $(0\ a\ W - 1)$, onde $L = 0$ representa o pixel preto e $L = W - 1$ representa o pixel branco. Normalmente W corresponde a uma potência inteira positiva de base 2. Como exemplo, para uma imagem de 8 bits, existem 256 níveis de cinza ($W = 2^8 = 256$).

Para imagens que possuem informações em mais de uma banda, chamadas de imagens multibanda, existe uma função $f(x, y)$ para cada banda. Por exemplo, imagens coloridas que utilizam padrão RGB são formadas por 3 bandas de cores primárias, vermelho (R), verde (G) e azul (B), onde cada pixel é representado pela mistura das 3 cores.

Para converter uma cena real em uma imagem digitalizada, duas etapas são imprescindíveis: a aquisição da imagem e sua digitalização.

2.1.1 Aquisição de imagens

Segundo Gonzales e Woods, 2000, dois elementos são necessários para aquisição de imagens, um dispositivo físico que seja sensível a uma banda do espectro da energia eletromagnética e que produza um sinal elétrico de saída proporcional ao nível de energia recebido, e um dispositivo para conversão desse sinal elétrico recebido para uma forma digital.

O processo de aquisição de imagens consiste em 3 etapas; (i) energia refletida pelo objeto de interesse, (ii) um sistema ótico que foca essa energia e (iii) um sensor que mede essa quantidade de energia [Moeslund,2012].

2.1.1.1 Fontes de energia

Para capturar uma imagem, a câmera precisa de uma fonte de energia capaz de iluminar a cena, e essa fonte deve emitir uma certa quantidade de energia que seja mensurável. Neste trabalho é considerada a luz visível como fonte de energia, conforme esquema apresentado na Figura 2.2 [Moeslund, 2012].

No processo de aquisição de imagem, a iluminação deve ser considerada como uma variável importante no processo. Segundo Moeslund, 2012, na área de visão de máquinas, a iluminação é muito mais importante do que, por exemplo, os softwares utilizados no projeto.

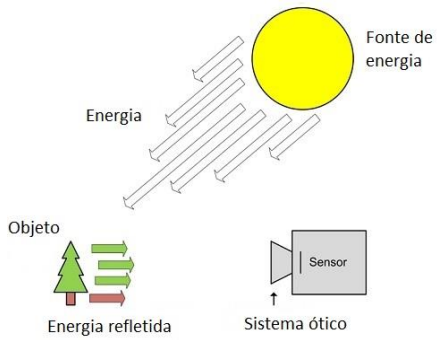


Figura 2.2 - Ilustração de um processo típico de aquisição de imagem

2.1.1.2 Sistema óptico

Com o objeto iluminado, a luz refletida por ele deve ser capturada pela câmera. Entretanto, a luz refletida de diversos pontos do objeto é misturada, gerando uma imagem sem uso. Ainda, luz refletida pelo entorno do objeto também é capturada, gerando um resultado ainda pior. A solução é colocar uma barreira entre o objeto de interesse e o elemento sensível que irá capturar a imagem. Como consequência, a imagem é capturada de cabeça para baixo, porém os softwares e hardwares utilizados atualmente normalmente rearranjam essa imagem para posição correta, conforme ilustra a Figura 2.3 [Moeslund, 2012].

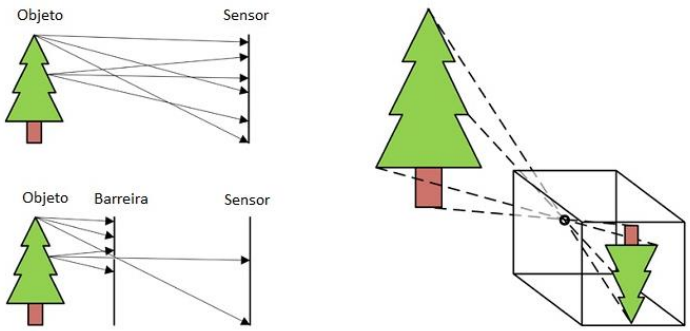


Figura 2.3 – Ilustração do sistema óptico de uma câmera.

O conceito de barreira é válido, porém pouca luz entra pelo orifício e atinge o sensor, por isso o orifício é substituído por um sistema óptico, onde as lentes são os principais elementos. As lentes são basicamente um objeto de vidro que tem a função de focar a luz que entra em direção ao sensor.

Na Figura 2.4 representa-se como as lentes focam a luz emitida pelo objeto. Na figura são ilustrados três raios de dois pontos distintos, que são refletidos pelo objeto em direção a

lente. Quando os raios incidem na lente, eles são direcionados para formar a imagem que é capturada pelo sensor da câmera, posicionado naquele local. Nota-se que os raios paralelos se interceptam em um ponto F, chamado de ponto focal. A distância do centro da lente, ponto O, até o ponto focal é chamada de distância focal *f*. A linha onde estão os pontos O e F é chamado de eixo ótico [Moeslund,2012].

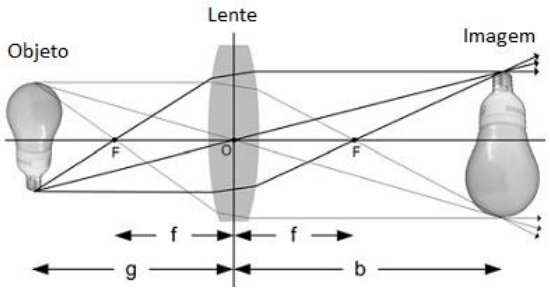


Figura 2.4 – Luz refletida pelo objeto sendo focada pela lente para formar a imagem.

2.1.1.3 Sensor de imagem

A luz refletida pelo objeto é focada pelo sistema ótico e precisa ser gravada pela câmera. Para isso um sensor é utilizado, que consiste em um arranjo 2D de células, onde cada célula é denominada pixel e é capaz de medir a quantidade de luz incidente e convertê-la em voltagem, que então é representada em um sinal digital, conforme Figura 2.5 [Moeslund, 2012].

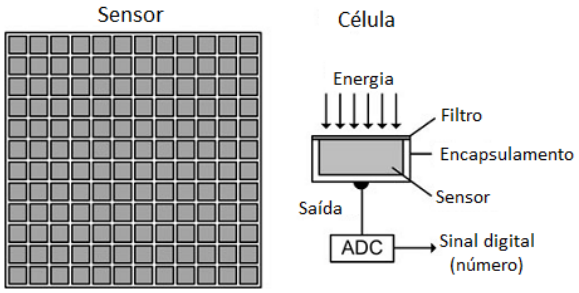


Figura 2.5 – Ilustração de um sensor de imagem de uma câmera digital.

Quando a câmera inicia o processo de captura da imagem, o obturador da câmera permite a entrada de luz e as cargas começam a se acumular nas células. Após um tempo, conhecido como tempo de exposição, o obturador da câmera bloqueia a entrada da luz e a imagem é então gerada. Se o tempo de exposição for muito longo, pode ocorrer a super-

exposição, que gera uma imagem muito clara, e se o tempo for muito curto, pode ocorrer a sub-exposição, gerando uma imagem muito escura. A maioria das câmeras possuem um sistema inteligente, chamado de controle automático de ganho, que tenta evitar o tempo de exposição incorreto, porém para sistemas que utilizam análise e processamento de imagem, esse sistema inteligente pode não ser tão atrativo, pois é necessário ter controle sobre a iluminação para poder processar a imagem corretamente. Outro problema que pode ser causado pelo tempo de exposição inadequado é o borramento da imagem, principalmente quando o objeto está em movimento.

As cargas acumuladas pelas células são convertidas pelo conversor analógico-digital (ADC – *analog-digital converter*) em um sinal digital. Esse processo de conversão é que torna a imagem em um arquivo digital.

Existem dois tipos de sensores de imagem utilizados em câmeras, o CCD (*Charge-coupled Device*) e o CMOS (*Complementary Metal-oxide Semiconductor*). Ambos os sensores recebem uma quantidade de luz em cada uma das células e convertem essa luz em um sinal elétrico. No sensor CCD, a quantidade de luz que incide nas células é convertida em sinal elétrico em um ou mais barramentos de saída, e enviada do sensor como um sinal analógico. Esse sinal de saída do sensor é então amplificado e convertido para um sinal digital em um conversor ADC. No sensor CMOS, a quantidade de luz que incide nas células é convertida em um sinal elétrico na própria célula, e normalmente o próprio sensor possui amplificador, correção de ruído e circuito de digitalização, fornecendo como saída do sensor um sinal digital. Esse sinal digital obtido como saída desses circuitos é que forma a imagem digital.

Na Figura 2.6 são ilustrados os dois tipos de sensores utilizados pelas câmeras, CCD (a) e CMOS (b).

(a) (b)

Figura 2.6 - Sensores de imagem.

2.1.2 Digitalização de Imagens

A digitalização da imagem inclui a amostragem (discretização espacial) e a quantização (discretização em amplitude).

A amostragem converte a imagem analógica, obtida pelos sensores na etapa de aquisição, em uma matriz de M por N pontos, denominados pixels. Quanto maior o valor de M e N, maior é a resolução da imagem e consequentemente maiores os detalhes captados pela imagem. Porém quanto maior a resolução, maior é o número de bytes necessários para armazená-la e mais capacidade computacional é necessária para processar essa imagem.

A Figura 2.7 ilustra a posição de um pixel na imagem, que é representada por um par de coordenadas *[c, r]*, onde *c* é a coluna e *r* a linha em que se encontra o pixel, sendo a origem do sistema a parte superior esquerda da imagem, e o pixel naquela posição tem as coordenadas [1, 1].

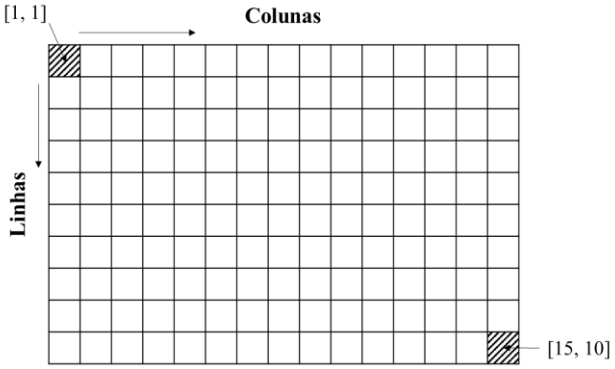
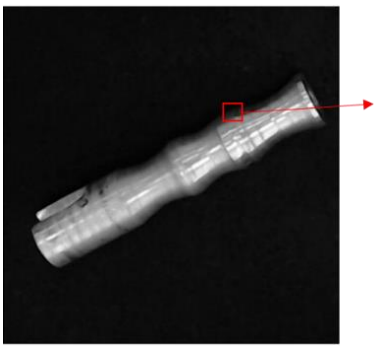


Figura 2.7 - Sistema de coordenadas nas imagens digitais

A quantização atribui a cada um dos pixels um valor inteiro, na faixa de 0 a $2^n - 1$, onde n é o número de bits que representa a cor em cada pixel da imagem. Conforme exposto anteriormente, imagens de 8 bits são representadas por 256 níveis de cinza, ou seja, cada pixel recebe um valor entre 0 e 255 que representa sua cor. Portanto, uma imagem de 8 bits representada em tons de cinza é um arranjo 2D de pixels, onde cada pixel possui um valor no intervalo de 0 a 255, conforme exemplo da Figura 2.8.

A resolução da imagem é equivalente ao tamanho do arranjo 2D que discretiza a imagem, por exemplo, para imagens com resolução 800x600 pixels, o arranjo 2D possui 800 colunas e 600 linhas, totalizando 480.000 pixels.



29	31	33	35	35	34	31	30	37	32	30	36
32	34	37	39	38	36	33	31	41	37	35	37
36	38	41	43	42	39	36	34	39	39	40	42
39	41	44	46	45	42	38	36	35	38	43	48
43	47	50	49	44	40	38	39	39	45	54	62
47	51	55	57	58	62	68	74	82	86	90	93
56	59	63	67	74	86	100	110	117	118	117	115
67	69	73	78	86	99	115	126	129	128	126	121
76	80	85	91	97	107	120	129	137	137	133	128
85	90	97	103	108	116	125	132	143	143	141	135
95	100	107	111	115	120	128	133	139	141	142	138
104	108	112	115	116	120	126	132	134	138	141	140
113	116	120	122	124	127	131	133	134	139	144	144

(a) (b) (c)

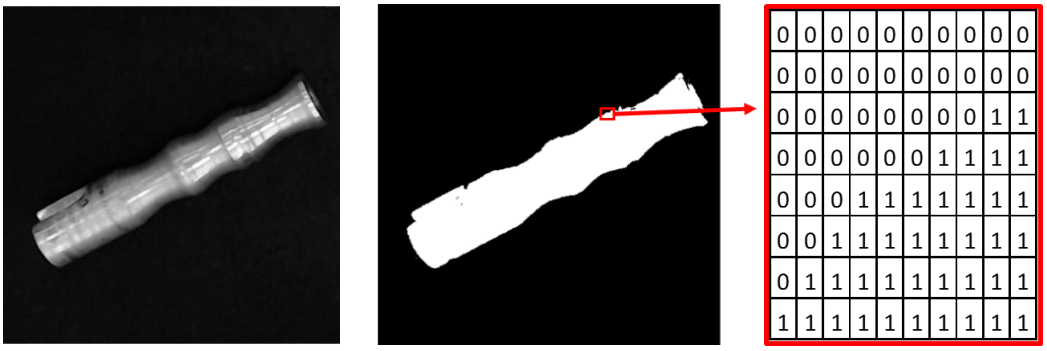
Figura 2.8 - Imagem em tons de cinza (a) e sua representação 2D (b) (c).

Segundo Marques Filho e Vieira Neto, 1999, 64 níveis de cinza são suficientes para o olho humano, entretanto sistemas de visão artificial normalmente utilizam imagens com 256 níveis de cinza.

2.1.2.1 Imagens binárias

Segundo Acharya e Ray, 2005, as imagens digitais podem ser (i) binárias, (ii) tons de cinza ou (iii) coloridas. Em imagens binárias, os pixels podem assumir somente dois valores, 0 ou 1, enquanto em imagens com tons de cinza podem sem quantizados em diversos níveis de intensidade, e em imagens coloridas podem ser quantizados em bandas de diferentes cores. Conforme o número de níveis de intensidade aumenta, a imagem representa a cena com melhor aproximação, porém os requerimentos para armazenamento do arquivo também crescem proporcionalmente. As imagens binárias são as que menos utilizam, tanto armazenamento quanto processamento.

Existem um grande número de aplicações para imagens binárias em visão computacional, como por exemplo reconhecimento de objetos e rastreamento, entretanto as aplicações são limitadas devido à pouca informação contida nesse tipo de imagem. Uma imagem em tons de cinza pode ser convertida em uma imagem binária através do processo de limiarização. Na Figura 2.9 é apresentada uma imagem de 8 bits em tons de cinza, uma imagem limiarizada utilizando o valor 20 e uma parte da matriz discretizada de uma pequena região da imagem.



(a) (b) (c)

Figura 2.9 - Imagem em tons de cinza (a), binária (b) e quantização da imagem (c).

2.2 Processamento de imagens

Os olhos, em conjunto com o cérebro humano, são capazes de extrair muito mais informações e detalhes de uma imagem do que seria possível descrever em um texto, e essa habilidade é que se deseja replicar com a visão computacional. Para alcançar esse objetivo, a câmera substitui os olhos e os programas de processamento de imagens e vídeos substituem o cérebro humano [Moeslund, 2012].

As primeiras aplicações de processamento de imagens digitais foram no melhoramento da qualidade de imagens capturadas, porém com o crescimento do poder computacional o número de aplicações foi aumentando. Atualmente o processamento de imagens é utilizado na astronomia, medicina, compressão de imagens, eventos esportivos, reabilitação, fotos em movimento, vigilância e fiscalização, indústria, controle de robôs, biometria, entre outras aplicações.

2.2.1 Segmentação de imagens

Segundo Shih, 2010, segmentação de imagens é um processo de classificação de pixels, que tem como objetivo extrair ou segmentar objetos ou regiões de uma imagem do seu plano de fundo (*background*). Deve-se ressaltar que não existe uma abordagem padrão para segmentação, pois diferentes partes da imagem podem servir de descritor para segmentação, e que cada segmento pode ser extraído da imagem de diferentes maneiras. Selecionar uma técnica apropriada para segmentação vai depender do tipo de imagem e qual a sua aplicação.

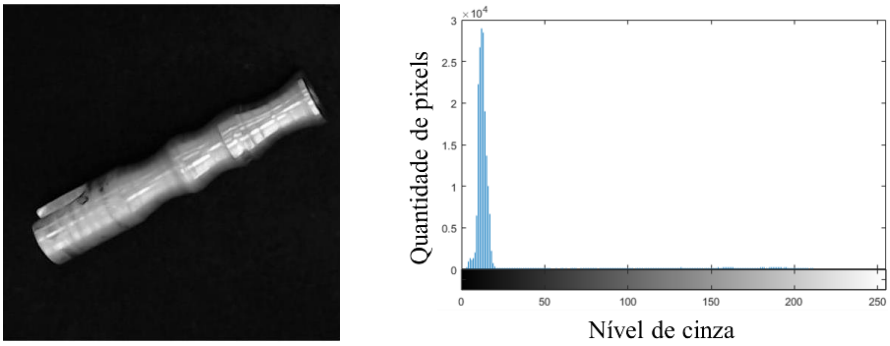
Ainda segundo Shih, 2010, há basicamente quatro tipos de técnicas de segmentação: (i) limiarização (*thresholding*), (ii) baseada em fronteiras (*boundary-based*), (iii) baseada em

região (*region-based*) e (iv) técnicas hibridas (que mistura detecção de bordas com técnica de crescimento de região). Neste trabalho será utilizada a limiarização como técnica de segmentação dos objetos.

2.2.1.1 Histograma de Imagem

O histograma de uma imagem é um conjunto de números indicando o percentual de pixels naquela imagem que apresentam um determinado nível de cinza. Estes valores são normalmente representados por um gráfico de barras que fornece para cada nível de cinza o número (ou o percentual) de pixels correspondentes na imagem. Através da visualização do histograma de uma imagem obtém-se uma indicação de sua qualidade quanto ao nível de contraste e quanto ao seu brilho médio (se a imagem é predominantemente clara ou escura) [Marques Filho e Vieira Neto, 1999].

A Figura 2.10 mostra uma imagem com resolução 451 x 451 pixels e 256 níveis de cinza, e seu respectivo histograma. Nota-se pelo histograma que há um predomínio da quantidade de pixels mais escuros, o que pode ser comprovado pela imagem, onde o plano de fundo é escuro e ocupa a maior parte da imagem.



(a) (b)

Figura 2.10 - Imagem do objeto (a) e seu histograma (b).

Técnicas de modificação de histograma podem ser aplicadas a imagem para alterar algumas de suas características, tais como o brilho ou o contraste da imagem, porém essas técnicas não serão abordadas neste estudo, visto que não se faz uso das mesmas no desenvolvimento do trabalho.

2.2.1.2 Limiarização (*Thresholding*)

O princípio da limiarização consiste em separar as regiões de uma imagem quando esta apresenta duas classes, o fundo e o objeto. Devido ao fato da limiarização produzir uma imagem binária, o processo também é denominado binarização. A forma mais simples de limiarização consiste na bipartição do histograma, convertendo os pixels cujo tom de cinza é maior ou igual a um certo valor de limiar T em brancos, e os demais em pretos [Marques Filho e Vieira Neto, 1999].

Supondo que o histograma da Figura 2.11 corresponde a uma imagem f(x,y), composta de um objeto claro e um fundo escuro, de modo que os pixels do objeto e do fundo tenham níveis de cinza agrupados em dois modos dominantes. A técnica mais simples de limiarização é particionar o histograma utilizando um único limiar global T. A imagem é varrida pixel por pixel e cada pixel é rotulado como objeto (branco) ou plano de fundo (preto), dependendo se o nível de cinza é maior ou menor que o limiar T [Gonzales e Woods, 2000].

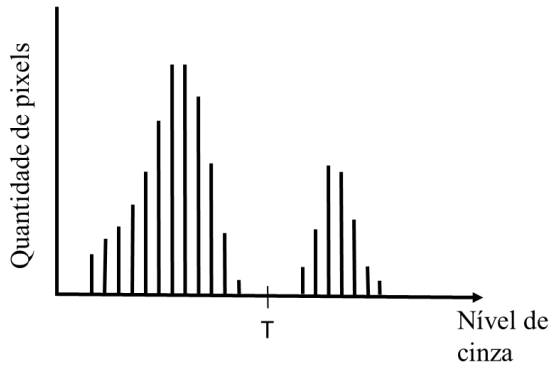


Figura 2.11 - Histograma de níveis de cinza.

2.2.1.3 Rotulagem de objetos (componentes conectados)

As técnicas de rotulagem de objetos baseados na conectividade dos pixels podem ser aplicadas em imagens binárias e imagens em tons de cinza. Existem diversos métodos e algoritmos desenvolvidos para realizar essa rotulagem. A técnica de rotulagem de objetos em uma imagem é aplicada quando há mais de um objeto e deseja-se extrai-los individualmente. Os objetos são representados como um agrupamento de pixels, onde cada agrupamento tem um rótulo único diferente de zero e o plano de fundo recebe o valor zero. Uma das técnicas requer somente duas varreduras. Para rotular os componentes com conectividade 4 (*4-connected*),

somente os pixels de cima e da esquerda são verificados. Se é utilizada a conectividade 8 (*8-connected*), os dois pixels da diagonal superior também são verificados, na análise de um pixel com seus vizinhos [Shih, 2010].

Depois de limiarizar uma imagem e identificar o que é plano de fundo e o que é objeto, a imagem pode ser segmentada em regiões utilizando um algoritmo de rotulagem de regiões conectadas, que resulta em regiões limitadas (*blobs*) que correspondem a cada objeto. O objetivo da rotulagem é determinar todos os agrupamentos de componentes conectados e atribuir o mesmo rótulo para cada pixel de um mesmo componente. Um método de rotulagem de componentes conectados é o método de dois passes. Para pixels com conectividade 4, no primeiro passe é atribuído um rótulo segundo o critério descrito a seguir [Acharya e Ray, 2005]:

(a) Se ambos os pixels, superior $P(i-1,j)$ e esquerdo $P(i,j-1)$, do pixel analisado $P(i,j)$ tem o mesmo rótulo X, então será atribuído ao pixel $P(i,j)$ o rótulo X;

(b) Se qualquer um dos pixels, superior $P(i-1,j)$ ou esquerdo $P(i,j-1)$, do pixel analisado tem o rótulo X, então será atribuído ao pixel $P(i,j)$ o rótulo X;

(c) Se o pixel superior $P(i-1,j)$ tem o rótulo X e o pixel da esquerda $P(i,j-1)$ tem um rótulo diferente Y ($X \neq Y$), então atribui X para o pixel $P(i,j)$. Coloca X e Y em uma tabela de equivalência.

(d) Se ambos os pixels, superior $P(i-1,j)$ e esquerdo $P(i,j-1)$, do pixel analisado $P(i,j)$ forem de rótulos diferentes, atribui um novo rótulo Z ao pixel $P(i,j)$.

A Figura 2.12 demonstra como é o arranjo de um pixel $P(i,j)$ com conectividade 4 e 8.

	$P(i-1,j)$	
$P(i,j-1)$	$P(i,j)$	$P(i,j+1)$
	$P(i+1,j)$	

(a)

$P(i-1,j-1)$	$P(i-1,j)$	$P(i-1,j+1)$
$P(i,j-1)$	$P(i,j)$	$P(i,j+1)$
$P(i+1,j-1)$	$P(i+1,j)$	$P(i+1,j+1)$

(b)

Figura 2.12 - Pixel com conectividade 4 (a) e conectividade 8 (b).

No segundo passe, os rótulos equivalentes são fundidos para criar um único rótulo para cada componente conectado da imagem. Se X e Y são dois rótulos equivalentes na tabela, então substitui Y por X, ou vice-versa, que resulta em um único rótulo para o componente [Acharya e Ray, 2005].

2.3 Morfologia matemática

A morfologia matemática estuda a estrutura geométrica das entidades presentes em uma imagem, podendo ser aplicada em várias áreas de processamento e análise de imagens, com objetivos tão distintos como realce, filtragem, segmentação, detecção de bordas, esqueletização, afinamento, entre outras. O princípio básico da morfologia matemática consiste em extrair as informações relativas à geometria e à topologia de um conjunto desconhecido (uma imagem), pela transformação através de outro conjunto completamente definido, chamado elemento estruturante [Marques Filho e Vieira Neto, 1999].

Morfologia matemática pode extrair características de forma da imagem, como bordas, filetes, cavidades, cantos, cunhas e fissuras, operando com vários elementos estruturantes modelados de acordo com a necessidade. Em aplicações de visão industrial, a morfologia matemática pode ser aplicada para implementar o reconhecimento rápido de objetos, aprimoramento de imagens, segmentação e inspeção de defeitos [Shih, 2010]. A Figura 2.13 ilustra três exemplos da utilização: remoção de pequenos objetos (a), preenchimento de furos (b) e separação de objetos (c) [Moeslund, 2012].

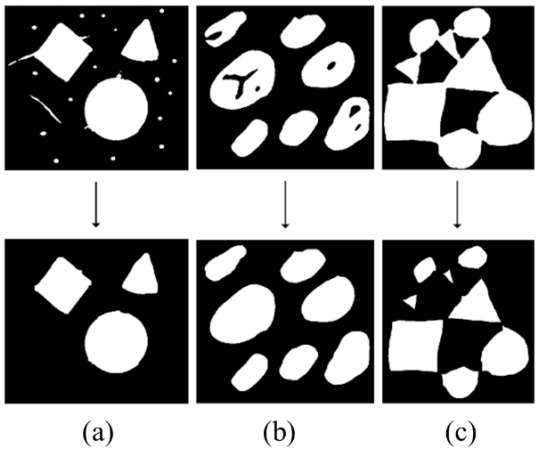


Figura 2.13 - Exemplos de uso da morfologia matemática

O elemento estruturante pode ser do tipo e do tamanho que o projetista desejar, porém geralmente um elemento com formato quadrado tende a preservar os cantos de objetos (Figura 2.14, parte superior), e elementos com formato de disco tendem a arredondar esses cantos (Figura 2.14, parte inferior) [Moeslund, 2012].

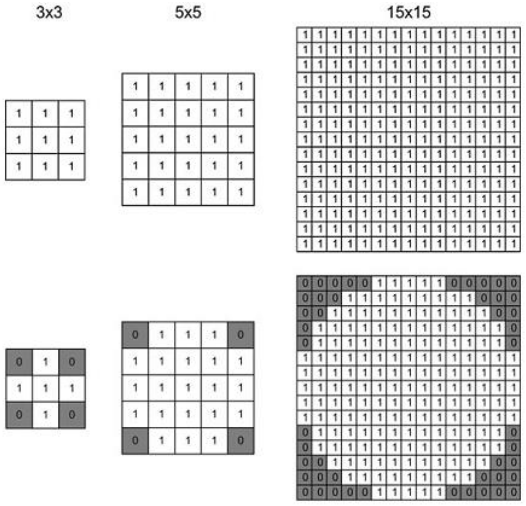


Figura 2.14 - Dois tipos de elementos estruturantes de diferentes tamanhos.

As operações morfológicas básicas são chamadas de Dilatação e Erosão, e podem ser combinadas, resultando em poderosas ferramentas de processamento de imagem conhecidas como operações compostas (abertura, fechamento, entre outras). Nesse trabalho é utilizada morfologia matemática aplicada somente a imagens binárias, porém também pode ser aplicada em imagens em tons de cinza e imagens coloridas.

2.3.1 Dilatação em imagens

O termo dilatação se refere ao fato de que o objeto da imagem binária irá aumentar de tamanho. Em geral, dilatar uma imagem resulta nos objetos ficarem maiores, pequenos furos serem preenchidos, e objetos serem fundidos ou tocarem-se. O tamanho dessas mudanças vai depender do tamanho do elemento estruturante [Moeslund, 2012].

A notação matemática que representa a dilatação é apresentada pela Equação 2.2, onde o símbolo $\oplus$ representa a dilatação.

$$g(x,y) = f(x,y) \oplus SE \qquad (2.2)$$

Onde:

$g(x,y)$ = imagem dilatada

$f(x,y)$ = imagem original

SE = elemento estruturante

A Figura 2.15 mostra o resultado de uma imagem binária dilatada pelo elemento estruturante SE. O elemento estruturante percorre todos os pixels da imagem, e quando o pixel

central do SE encontra um pixel com o mesmo valor na imagem original, aos pixels da vizinhança são atribuídos os valores iguais aos do SE, e assim gera a imagem dilatada [Moeslund, 2012].

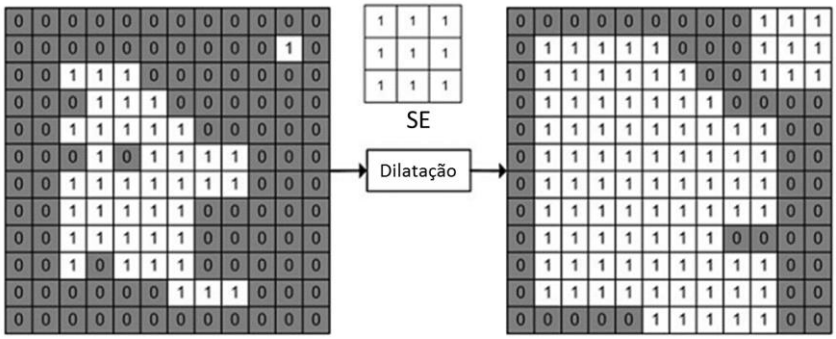


Figura 2.15 - Dilatação da imagem binária utilizando o elemento estruturante SE.

2.3.2 Erosão em imagens

O termo erosão se refere ao fato de que o objeto contido na imagem binária tem seu tamanho reduzido. Erosão em uma imagem, geralmente resulta em objetos menores, pequenos objetos desaparecerem, e objetos grandes se dividindo em outros menores. Assim como na dilatação, o efeito resultante depende do tamanho do elemento estruturante [Moeslund, 2012].

A notação matemática que representa a erosão é apresentada na Equação 2.3, onde o símbolo $\ominus$ representa a erosão.

$$g(x,y) = f(x,y) \ominus SE \qquad (2.3)$$

Onde:

$g(x,y)$ = imagem erodida

$f(x,y)$ = imagem original

SE = elemento estruturante

A Figura 2.16 mostra o resultado da erosão de uma imagem binária utilizando o elemento estruturante SE. O elemento estruturante percorre todos os pixels da imagem, e quando o pixel central do SE encontra um pixel com o mesmo valor, verifica se a vizinhança possui os mesmos pixels do SE, em caso afirmativo mantém o pixel central e muda os vizinhos para 0, em caso negativo muda para 0 todos os pixel, incluindo o pixel central. O elemento SE vai para o próximo pixel da imagem e verifica novamente, até percorrer toda imagem [Moeslund, 2012].

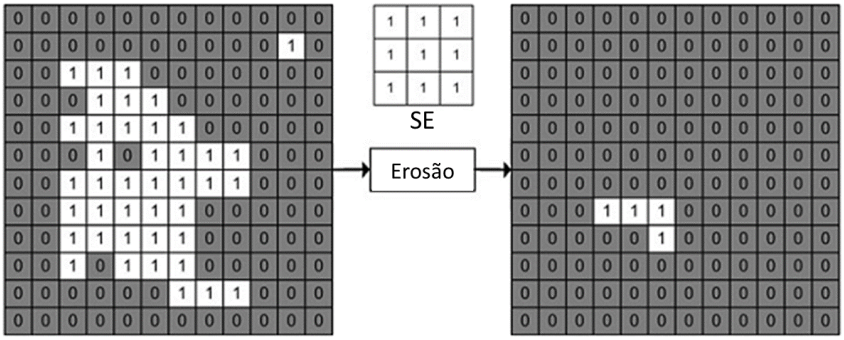


Figura 2.16 - Erosão da imagem binária utilizando o elemento estruturante SE.

2.4 Robôs Industriais

Segundo Siciliano et al., 2009, a robótica preocupa-se com o estudo das máquinas que podem substituir os seres humanos na execução de tarefas, tanto fisicamente como na tomada de decisões. Robótica é comumente definida como a ciência que estuda a conexão inteligente entre a percepção e a ação. Um sistema robótico é um sistema complexo, que pode ser representado funcionalmente por múltiplos subsistemas, conforme Figura 2.17 [Adaptado de Siciliano et al., 2009].

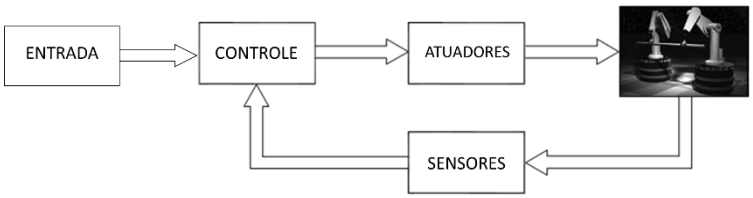


Figura 2.17 - Componentes de um sistema robótico.

Ao longo dos séculos, diversas invenções propiciaram o avanço tecnológico necessário para a gradual substituição do homem pela máquina. O avanço tecnológico das últimas décadas, passando pela criação das máquinas a vapor e pela produção em série de Henry Ford, teve reflexo direto na busca da redução de custos nos processos industriais, através da adoção de diversos modelos de produção, onde se destacam a automação programável e a automação flexível. Os robôs industriais têm crescente aplicação nos processos de automação programável e flexível, pois são máquinas capazes de executar diversos movimentos programados, adaptando-se as necessidades operacionais [Romano, 2002].

Segundo a norma ISO 10218-1 robô industrial é um manipulador multifuncional

reprogramável, automaticamente controlado, programável em três ou mais eixos que pode ser fixo no local ou móvel para uso em aplicações de automação industrial. Segundo Romano, 2002, um robô industrial é formado pela integração dos seguintes componentes:

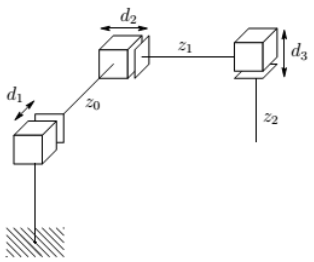
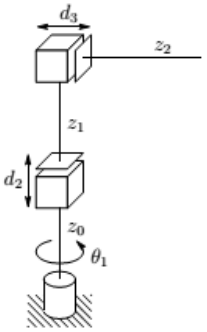
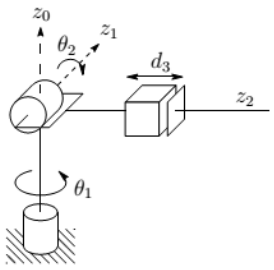
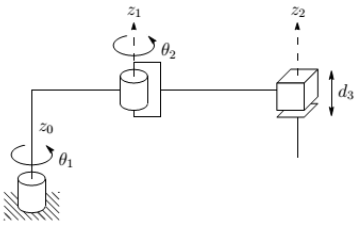
a) **Manipulador mecânico**: refere-se principalmente ao aspecto mecânico e estrutural do robô. É formado pelos elos (*links*) e juntas (*joint*), e sistema de transmissão de potência.
b) **Atuadores**: componentes que convertem energia elétrica, hidráulica ou pneumática em potência mecânica, que através dos sistemas de transmissão é enviada aos elos para que se movimentem.
c) **Sensores:** fornecem parâmetros sobre o comportamento do manipulador, geralmente em termos de posição e velocidade dos elos em função do tempo.
d) **Unidade de controle**: gerencia e monitora os parâmetros operacionais do robô. Os comandos enviados aos atuadores são originados de controladores de movimento e baseados em informações dos sensores.
e) **Unidade de potência**: responsável pelo fornecimento de potência necessária para a movimentação dos atuadores. A bomba hidráulica, o compressor e a fonte elétrica são exemplos de unidades de potência.
f) **Efetuador**: é o elemento de ligação entre o robô e o meio que o cerca. Pode ser uma garra, que basicamente pega o objeto e transporta-o até outra posição, ou uma ferramenta, que tem a função de realizar uma tarefa sobre a peça, sem necessariamente manipulá-la.

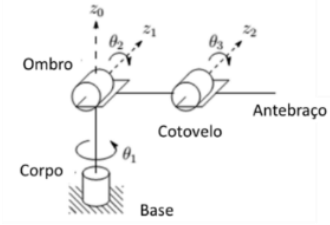
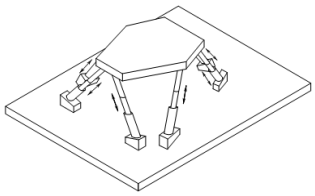
Segundo Siciliano et al., 2009, a estrutura mecânica de um manipulador robótico consiste na sequência de corpos rígidos, os elos, interconectados por articulações, as juntas. O manipulador é caracterizado por um braço (*arm*) que garante mobilidade, um punho (*wrist*) que confere destreza, e um efetuador (*end-effector*) que realiza as tarefas do robô. A mobilidade do manipulador robótico é garantida pelas juntas, que podem ser prismáticas, as quais promovem movimento relativo de translação ente dois elos, ou juntas de revolução, que promovem movimento relativo de rotação entre dois elos. Em uma cadeia cinemática aberta, cada junta agrega um grau de liberdade à estrutura (DOF – *degree of freedom*). Assim, em uma tarefa que consiste em posicionar e orientar um objeto em um espaço 3D, seis DOF são necessários, três para posicionar o objeto e três para orientá-lo.

O tipo e a sequência dos graus de liberdade do braço robótico, a partir da junta da base

do manipulador, permite a classificação dos manipuladores como cartesiano, cilíndrico, esférico, SCARA e antropomórfico, conforme indicado na Tabela 2.1 [Spong et al., 2006]. Existe ainda robôs de cadeia cinemática fechada, como os robôs paralelos.

Tabela 2.1 - Classificação dos manipuladores robóticos

Cartesiano	
	Formado por três juntas prismáticas, permitindo tipicamente movimento segundo três eixos mutualmente ortogonais.
Cilindrico	
	A primeira junta é de revolução e proporciona a rotação em torno da base, e as outras duas juntas são prismáticas.
Esférico	
	A primeira e a segunda juntas são de revolução, e a terceira junta é prismática
SCARA	
	É uma configuração especial, onde utiliza-se duas juntas de revolução e uma junta prismática de maneira que todos os eixos de movimento sejam paralelos. Este robô oferece alta rigidez para movimentação de cargas e é utilizado em tarefas de montagem de componentes de pequenas dimensões.

Antropomórfico	
	A configuração é caracterizada por três juntas de revolução, onde o eixo da primeira junta é ortogonal aos outros dois eixos de revolução, que são paralelos entre si.
Paralelo	
	Possui duas ou mais cadeias cinemáticas independentes conectando a base ao efetuador.

As principais indústrias que utilizam robôs para executar tarefas são a indústria automotiva, seguida pela indústria eletro-eletrônica, indústria metal-mecânica e de máquinas, indústria de plásticos e borracha, e alimentícia.

Segundo a Federação Internacional de Robótica (IFR), em 2016 a venda de robôs aumentou 16% comparado ao ano anterior, com destaque para a indústria eletro-eletrônica, onde houve aumento de 41%. A indústria automotiva continua sendo o principal cliente com uma fatia de 35% das vendas, seguida pela indústria eletro-eletrônica com 31% das vendas. Os países que mais compraram foram a China, Coréia do Sul, Japão, Estados Unidos e Alemanha, que representam 74% das vendas.

2.5 Redes Neurais Artificiais

A origem da inteligência artificial é datada dos anos 1930-1940, e desde então alcançou grandes avanços e ao mesmo tempo enfrenta grandes dificuldades. O principal objetivo das pesquisas relacionadas a inteligência artificial é utilizar os modelos computacionais para simular o comportamento inteligente dos seres humanos ou animais, simular a estrutura do cérebro e suas funções e entender o processo do pensamento humano. Um sistema de inteligência artificial geralmente deve ser capaz de realizar três tarefas [He e Xu, 2007]:

a) representar e armazenar conhecimento;
b) resolver vários problemas utilizando o conhecimento armazenado;

c) adquirir novos conhecimentos quando o sistema estiver funcionando (ter capacidade de aprendizagem).

Em 1943 o fisiologista McCulloch e o matemático Pitts desenvolveram o primeiro modelo matemático de um neurônio artificial, imitando o mecanismo de processamento de informação de um neurônio biológico, e esse desenvolvimento marcou o início das pesquisas sobre redes neurais artificiais baseadas no conexionismo. Esse modelo, ilustrado na Figura 2.18 implementa um classificador linear, onde a saída é obtida quando a combinação linear das entradas ultrapassa um limiar dado pela função de ativação [Du e Swamy, 2014].

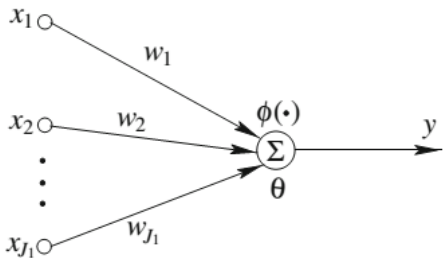


Figura 2.18 - Modelo matemático simples de um neurônio.

Nesse modelo de neurônio, a saída y é uma combinação das entradas x_{j_1} e dos pesos w_{J_1}, com um limiar dado por θ (*bias*) e uma função de ativação $\emptyset(\cdot)$ (Equações 2.4 e 2.5).

$$rede = \sum_{i=1}^{j_1} w_i\, x_i - \theta \qquad (2.4)$$

$$y = \emptyset(rede) \qquad (2.5)$$

Em 1958, Rosenblatt introduziu o conceito do perceptron, que do ponto de vista da engenharia foi o primeiro modelo de rede neural artificial (RNA) aplicado em processamento de informações. O modelo do perceptron é simples, e tem como características o armazenamento distribuído, processamento paralelo, habilidade de aprendizado, computação contínua [He e Xu, 2007].

As redes neurais são formadas por nós ou unidades (neurônios) conectados por ligações direcionadas. Cada ligação tem um peso numérico w_i associado a ela, que determina a força e o sinal da conexão, onde a saída primeiro calcula uma soma ponderada das entradas ($w_i x_i$), e

em seguida aplica a função de ativação, que tipicamente tem um limiar rígido ou uma função limiar. Uma vez definido o modelo matemático dos neurônios individuais, eles são conectados para formar uma rede neural, que é caracterizada pela sua arquitetura, características dos nós e regras de aprendizado. Generalizando, existem duas formas de conectar os neurônios: rede com alimentação para a frente, rede recorrente, ou um misto das duas.

As funções de ativação podem ser do tipo (i) função de etapa binária, que na prática não é utilizada, (ii) função linear simples, (iii) função sigmoide, (iv) função tangente hiperbólica, (v) função ReLU, (vi) Leaky ReLU e (vii) softmax, que são funções não lineares. Nesse trabalho não serão abordadas as características, vantagens e desvantagens de cada uma dessas funções pois não está dentro do escopo do estudo.

As redes com alimentação para a frente (*feedforward neural networks*) tem conexões em somente uma direção, ou seja, cada nó (neurônio) recebe como entrada a saída dos nós anteriores e sua saída está conectada ao próximo nó, não havendo laços. Uma rede recorrente (*recurrent neural network*) alimenta suas saídas de volta as suas próprias entradas, e isto significa que é uma rede dinâmica que pode atingir um estado estável, ou apresentar oscilações ou até um comportamento caótico.

Segundo Du e Swamy, 2014, as topologias mais populares para redes neurais são as (i) redes com alimentação para a frente em camadas totalmente conectadas (*fully connected layered feedforward network*), (ii) rede recorrente (*recurrent network*), (iii) rede em treliça (*lattice network*), (iv) rede com alimentação para a frente em camadas com conexões laterais (*layered feedforward networks with lateral connections*) e (v) redes celulares (*cellular network*).

Na Figura 2.19 são apresentadas arquiteturas de redes neurais com (a) alimentação para a frente em camadas totalmente conectadas, (b) recorrente, (c) em treliça 2D, (d) com alimentação para a frente em camadas com conexões laterais e (e) celular, onde os círculos grandes numerados são neurônios e os círculos menores são nós de entrada [Du e Swamy, 2014]. Muitas vezes um neurônio, mesmo com várias entradas, pode não ser suficiente para resolver o problema. Pode ser necessário mais neurônios operando em paralelo, e a isso se dá o nome de camadas (*layers*). A Figura 2.20 representa uma rede neural com alimentação pra frente de camada simples com *m* neurônios, onde cada entrada $x_1, x_2, \dots, x_n$ está conectada em cada um dos neurônios, cada uma das conexões de entrada possui seus respectivos pesos, e obtêm-se as saídas $y_1, y_2, \dots, y_m$. É comum o número de entradas *n* ser diferente do número de

neurônios *m* [Hagan et al., 1996].

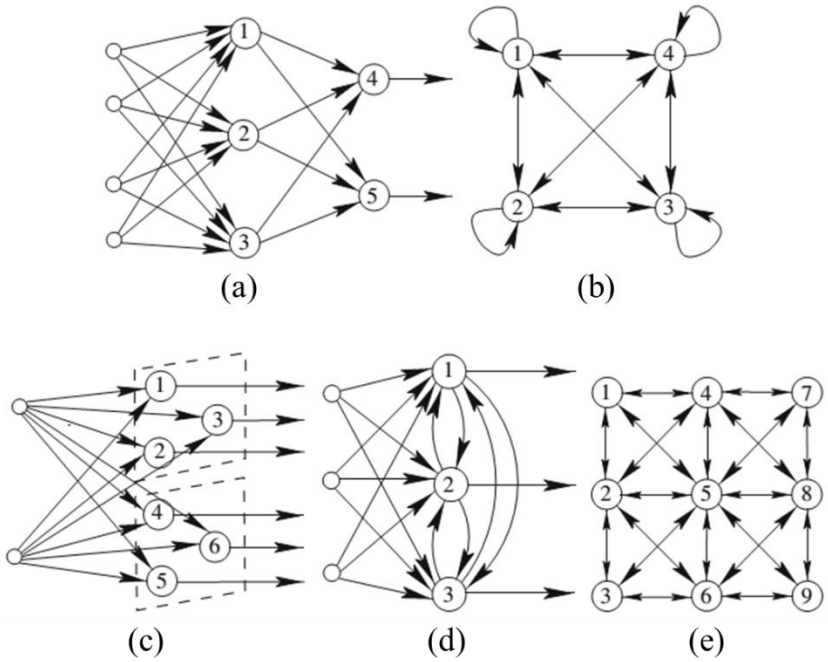


Figura 2.19 - Arquitetura de redes neurais.

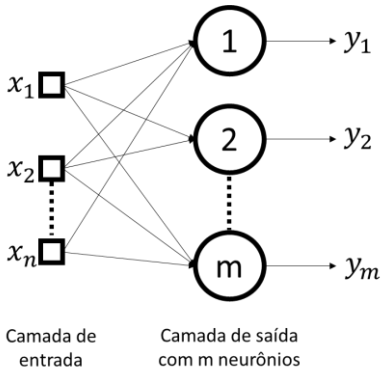


Figura 2.20 - Rede neural de camada simples

Outras redes utilizam mais de uma camada de neurônios, chamadas redes neurais multi-camadas, onde as camadas de neurônios são organizadas em sequência, sendo a primeira correspondente as entradas, que dependendo da abordagem poderá ou não ser considerada como uma camada a ser contada, camadas intermediárias que são chamadas de "escondidas", e a camada de saída. A Figura 2.21 mostra uma rede neural com alimentação para frente multi-camadas, com 3 camadas, sendo duas delas escondidas e a camada de saída.

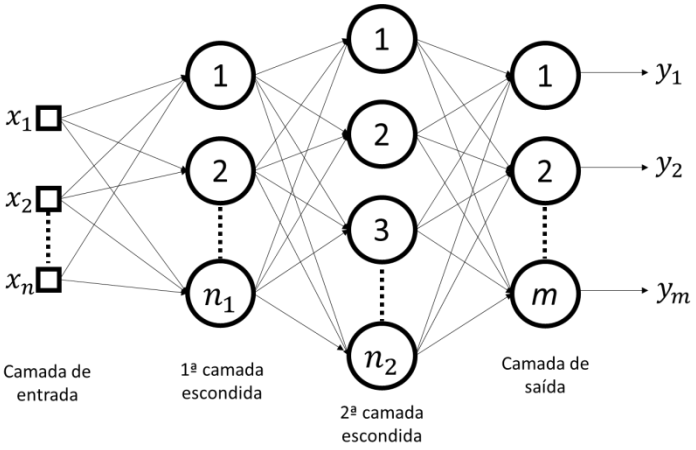


Figura 2.21 - Rede neural multi-camada de 3 camadas

A operação das redes neurais é dividida em dois estágios: aprendizado ou treinamento, e generalização. O aprendizado é uma das capacidades fundamentais das redes neurais, e as regras de aprendizado são algoritmos que encontram valores para os pesos e/ou outros parâmetros da rede. Segundo Du e Swamy, 2014, os métodos de aprendizado são divididos em supervisionado, sem supervisão e aprendizagem de reforço.

A aprendizagem supervisionada ajusta os parâmetros da rede através de uma comparação entre a saída atual da rede e a saída que seria desejada, ou seja, é um sistema de ciclo fechado com feedback (realimentação), onde o sinal de feedback (realimentação) é o erro. O valor do erro demonstra a diferença entre a saída atual da rede neural e a saída que foi obtida durante o treinamento.

Na Figura 2.22, conceitualmente, o "professor" tem conhecimento sobre o ambiente, adquirido por um conjunto de exemplos de entrada-saída, porém o ambiente é desconhecido pela rede neural de interesse. Supondo que o professor e a rede neural sejam expostos a um vetor de treinamento retirado do ambiente, o professor é capaz de fornecer à rede neural uma resposta desejada para aquele vetor de treinamento, devido ao seu conhecimento prévio, onde essa resposta desejada representa a ação ótima a ser realizada pela rede neural. Os parâmetros da rede são então ajustados sob a influência tanto do vetor de treinamento como do sinal de erro, e esse ajuste é realizado iterativamente, com o objetivo de fazer a rede neural emular o professor, ou seja, o conhecimento do ambiente disponível ao professor é transferido a rede neural através do treinamento, e uma vez que isso aconteça, pode-se dispensar o professor e deixar a rede neural lidar com o ambiente de forma independente [Haykin, 2001].

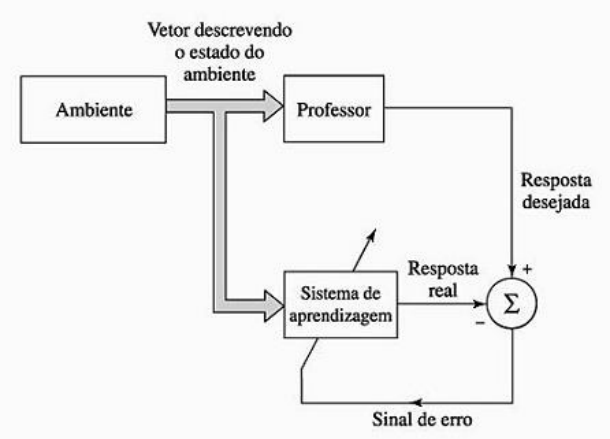


Figura 2.22 - Diagrama de blocos da aprendizagem supervisionada

Na aprendizagem sem supervisão não existem valores-alvo. Nela tenta-se associar informações das entradas, com uma redução intrínseca da dimensionalidade dos dados ou do tamanho total dos dados de entrada. A aprendizagem não supervisionada é baseada somente nas correlações entres os dados de entrada, e é usada para encontras padrões ou características significantes nesse conjunto de dados sem a ajuda de um professor externo. É necessário adotar um critério de parada nesse processo de aprendizagem, caso contrário o processo continua mesmo após encontrar um padrão [Du e Swamy, 2014]. A Figura 2.23 mostra um diagrama desse método de aprendizagem [Haykin, 2001].

Figura 2.23 - Diagrama de blocos da aprendizagem sem supervisão.

A aprendizagem de reforço é uma classe dos algoritmos computacionais que especifica como um agente artificial pode aprender a selecionar ações para maximizar o total do prêmio esperado. Pode-se dizer que esse tipo de aprendizagem é um caso especial da aprendizagem supervisionada onde a saída exata desejada é desconhecida. Nesse procedimento de aprendizagem a rede neural é recompensada por um "bom" resultado na saída ou é penalizada por um "mau" resultado na saída.

A aprendizagem supervisionada é amplamente usada em classificação, aproximação, controle, modelagem, identificação, processamento de sinais, e otimização. A aprendizagem sem supervisão é utilizada principalmente para agrupamento, quantização vetorial, extração de características, codificação de sinais, e análise de dados. O aprendizado de reforço é tipicamente utilizado em controle e inteligência artificial [Du e Swamy, 2014].

A generalização, em termos matemáticos, é a interpolação e extrapolação dos dados de entrada. O objetivo de uma rede neural não é aprender a exata representação dos dados de treinamento, mas construir um modelo estatístico do processo que gerou os dados. Quando uma rede neural é treinada com exemplos demais, ele pode apresentar bons resultados com os dados de treinamento, mas terá baixa capacidade de generalização para outros dados, o que acabará gerando resultados não desejáveis. Geralmente a capacidade de generalização de uma rede neural é determinada conjuntamente pelo (i) tamanho do conjunto de dados de treinamento, (ii) pela complexidade do problema e (iii) pela arquitetura da rede [Du e Swamy, 2014].

As RNA vêm sendo implementadas para resolver uma variedade de problemas, entre eles reconhecimento de voz, reconhecimento de letras escritas a mão, reconhecimento de digital, identificação submarina através de sonar, classificação meteorológica, controle automático de veículos, diagnóstico de hipertensão, detecção de anomalias no coração, identificação de fraude bancária, previsão de movimento em bolsa de valores, controle de processo, validação de dados, entre outros.

Mishra e Srivastava, 2014, apontam algumas limitações das RNA, tais como: as redes neurais não conseguem resolver problemas do dia-a-dia, não existe uma metodologia estrutural disponível para RNA, a qualidade da saída de uma RNA pode ser imprevisível, muitos sistemas de RNA não descrevem como resolvem os problemas, tem natureza de caixa preta, grande exigência computacional, e suscetível a gerar mais resultados que o necessário (*overfitting*).

O Anexo I apresenta algumas das arquiteturas de rede neural, porém novas arquiteturas são criadas de tempos em tempos, com novas estruturas de nós, regras de aprendizagem, etc. Apesar de algumas das arquiteturas presentes no anexo serem aparentemente iguais, existem processos de treinamento e regras de aprendizagem que podem ser diferentes.

No presente trabalho é adotada a arquitetura de rede com alimentação para frente totalmente conectada, com uma camada escondida e uma camada de saída, implementada em ambiente MATLAB®, para aplicação específica na determinação da orientação de objetos (ver Anexo I).

2.6 Comunicação e transmissão de informações

Atualmente as pessoas utilizam computadores para realizar chamadas, jogar com outras pessoas, utilizar aplicativos de mensagem instantânea, assistir TV, e comprar quase tudo que se possa imaginar, e isso tudo é possível devido a capacidade dos programas de se comunicarem através da internet. Para ocorrer essa comunicação, existem redes de computadores, que consistem em maquinas interconectadas e com um canal de comunicação definido, pacotes de informações que são solicitados e enviados através da rede, e um protocolo que define as regras dessa comunicação [Donahoo e Calvert, 2009].

A internet é um exemplo de rede de computadores que interconecta centenas de milhões de dispositivos ao redor do mundo, entre eles estão os computadores, TVs, smartphones, consoles de jogos, sistemas de segurança, automóveis, entre outros. Nessa rede, pacotes são enviados e recebidos, contendo dados e informações para que o dispositivo final possa recebe-los e interpretá-los de maneira correta. "Um protocolo define o formato e a ordem das mensagens trocadas entre duas ou mais entidades comunicantes, bem como as ações realizadas na transmissão e/ou no recebimento de uma mensagem ou outro evento" [Kurose e Ross, 2013].

Implementar uma rede funcional requer que diversos problemas sejam resolvidos, e para resolvê-los uma série de protocolos são utilizados, sendo o TCP/IP um desses conjuntos de soluções para os problemas, também chamado de pacote de protocolos, que além de ser utilizado na internet, pode ser utilizado em redes privadas locais. Os principais protocolos do pacote TCP/IP são o Protocolo de Internet (IP - *Internet Protocol*), o Protocolo de Controle de Transmissão (TCP - *Transmission Control Protocol*) e o Protocolo de Datagrama de Usuário (UDP - *User Datagram Protocol*) [Donahoo e Calvert, 2009].

O pacote de protocolos TCP/IP permite que computadores, smartphones, e outros dispositivos, de diferentes fabricantes e rodando diferentes aplicações, se comuniquem apropriadamente [Fall e Stevens, 2012].

O protocolo IP oferece comunicação lógica entre dois dispositivos, oferecendo um serviço de entrega de melhor esforço, ou seja, ele faz o "melhor esforço" para entregar os pacotes entre dois sistemas que se comunicam, porém não garante que esses pacotes serão entregues, estarão íntegros ou em ordem, tornando-o um serviço não confiável. Cada dispositivo tem, no mínimo, um endereço de rede, chamado endereço IP [Kurose e Ross, 2013].

Os protocolos TCP e UDP são chamados de protocolos de transporte *end-to-end*, onde

os pacotes são transportados de uma aplicação até outra, por todo caminho, diferentemente do protocolo IP que transporta os pacotes somente entre dois dispositivos. O TCP e o UDP têm uma funcionalidade em comum, o endereçamento, enquanto o protocolo IP entrega pacotes para dispositivos, um endereço mais completo é necessário para entregar esses pacotes a uma aplicação em particular de um determinado dispositivo, assim ambos protocolos utilizam informações adicionais chamadas de número da porta (*port number*), para identificar aplicações dentro de um mesmo dispositivo [Donahoo e Calvert, 2009].

O protocolo TCP é projetado para detectar e recuperar pacotes perdidos, duplicados, ou qualquer outro erro que pode ocorrer na comunicação entre dispositivos promovida pelo protocolo IP. O TCP é um protocolo com conexão orientada, onde antes do envio de pacotes se estabelece uma conexão TCP, que envolve completar uma troca de mensagem de apresentação (*hanshake message*) entre a implementação TCP dos dois aplicativos que se comunicarão. O protocolo TCP oferece um serviço de transferência confiável de dados, que garante que os dados sejam entregues da aplicação remetente para a aplicação destino corretamente e em ordem, transformando um serviço de transporte de dados não confiável entre dispositivos, como o IP, em um serviço confiável entre aplicações.

O protocolo UDP é muito mais simples, ele não garante que os pacotes cheguem ao destino final, podendo haver perdas de pacotes, reordenamento, porém a integridade dos dados é garantida. Esse protocolo não requer uma conexão estabelecida entre as duas aplicações para enviar os pacotes, ele cria o pacote de dados e envia com o endereçamento através do protocolo IP para o destinatário, sem necessidade de conexão.

Neste trabalho será utilizada a comunicação via protocolo TCP entre o software de desenvolvimento do método e o software conectado ao controlador do robô.

2.6.1 Comunicação entre aplicações através de *Socket* TCP/IP

A conexão TCP é um serviço *full-duplex*, onde os dados podem trafegar de uma aplicação A para B ao mesmo tempo que os dados trafegam de B para A, e é uma conexão sempre ponto-a-ponto, entre um único remetente e um único destinatário [Kurose e Ross, 2013].

A comunicação entre as duas aplicações acontece de forma que primeiramente uma delas envia uma mensagem, denominada cliente, enquanto a outra aguarda, denominada servidor. A diferenciação entre cliente e servidor é importante, pois o cliente precisa conhecer o endereço do servidor e o número da porta de comunicação, e o servidor somente seu próprio

endereço, e ser configurado com a mesma porta configurada no cliente [Donahoo e Calvert, 2009]. Para estabelecer a comunicação, o cliente envia um segmento TCP especial, o servidor responde com um segundo segmento TCP especial, e por fim o cliente responde novamente com um terceiro segmento TCP especial, estabelecendo a conexão. Esse procedimento é conhecido como apresentação de 3 vias (*3-way handshake*), e uma vez estabelecida a conexão TCP, as aplicações podem enviar dados uma para outra [Kurose e Ross, 2013].

Uma conexão TCP consiste em 2 pares de terminações ou *sockets*, um par em cada aplicação, onde cada *socket* é identificado por um par que contém um endereço IP e um número de porta para comunicação. A conexão normalmente tem 3 fases: configuração, transferência de dados e encerramento da conexão [Fall e Stevens, 2012].

O *socket* é um meio através do qual é possível enviar e receber dados, permitindo que uma aplicação conecte em uma rede e se comunique com outra aplicação que também está conectada a essa rede. Um *socket* TCP/IP é identificado por um único endereço IP, por um protocolo *end-to-end* (TCP ou UDP) e por um número de porta [Donahoo e Calvert, 2009].

Quando a conexão TCP é estabelecida entre duas aplicações, é possível enviar e receber dados. Nesse processo o cliente envia uma cadeia de dados através do *socket* para o servidor, após os dados passarem pelo *socket* eles são direcionados pelo protocolo TCP para um *buffer* de envio, de onde são enviados pequenos segmentos de dados através da rede para o servidor. O protocolo TCP atribui um cabeçalho a esses pequenos segmentos, contendo diversas informações que garantirão o envio e recebimento dos dados corretamente, formando o chamado segmento TCP, que é composto por um campo de cabeçalho e um de dados (que contém uma parte dos dados a serem enviados). Esses segmentos são passados para a camada de rede e encapsulados separadamente dentro dos datagramas IP (Figura 2.24), sendo enviados até o servidor e armazenados no *buffer* de recepção, onde serão lidos e interpretados pela aplicação, reconstruindo os dados originais enviados pelo cliente, conforme ilustra a Figura 2.25 [Kurose e Ross, 2013].

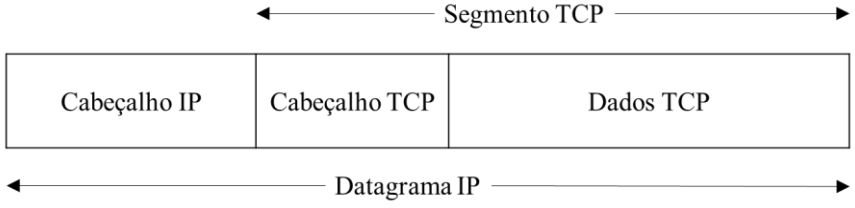


Figura 2.24 - Segmentos TCP encapsulados em datagramas IP

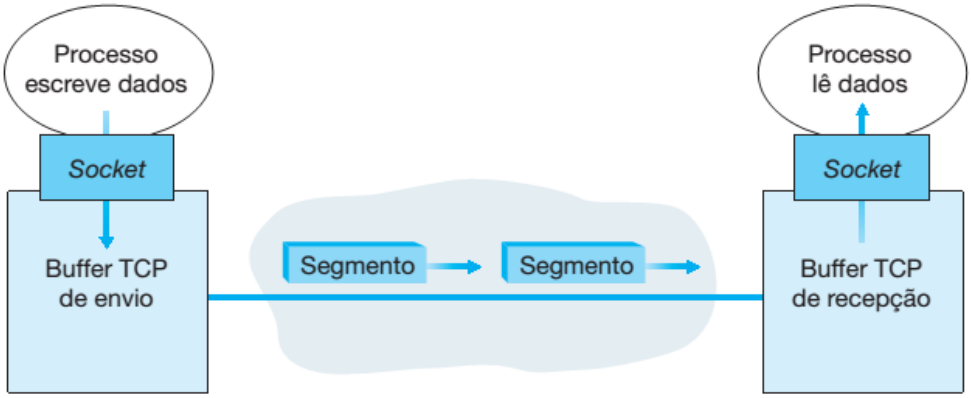


Figura 2.25 - *Buffers* TCP de envio e recepção

Uma aplicação no servidor ou no cliente pode ter diversos *sockets* em uso ao mesmo tempo, assim como múltiplas aplicações podem utilizar o mesmo socket, embora seja menos usual [Donahoo e Calvert, 2009].

Os segmentos TCP enviados pela rede utilizam um número sequencial, presente no campo de cabeçalho, que os identifica, possibilitando ao recebedor descartar pacotes duplicados e reordená-los corretamente para serem lidos pela aplicação destino, sem prejuízo aos dados enviados. O protocolo TCP também possibilita o controle de tráfego de dados entre as duas aplicações, permite determinar um tempo de espera (*timeout*) para receber a confirmação da entrega dos segmentos no destino, e possui um método de somatório de verificação entre as duas aplicações para garantir a troca confiável de dados [Fall e Stevens, 2012].

3 DESENVOLVIMENTO DO TRABALHO

A implementação desse trabalho tem o objetivo principal de permitir, por meio de um sistema de visão, reconhecer objetos presentes na área de atuação de um robô, limitada pelo campo visual de uma câmera.

A infraestrutura de desenvolvimento envolve equipamentos e acessórios do Laboratório de Automação e Robótica do Departamento de Engenharia Mecânica da UFRGS.

O reconhecimento das imagens de componentes no escopo do trabalho implica no desenvolvimento de um sistema computacional que permite a identificação do objeto, a determinação das coordenadas do respectivo baricentro, bem como sua orientação em relação a um sistema de referência.

3.1 Equipamentos e software utilizados

3.1.1 Manipulador Robótico

O sistema de visão foi desenvolvido para ser utilizado conjuntamente com o manipulador Epson SCARA Série G6, que utiliza o controlador modelo C180. O manipulador é formado por 2 juntas de revolução (juntas 1 e 2) e uma junta prismática (junta 3), conforme explicado na seção 2.4, e por outra junta de revolução que realiza o movimento angular do efetuador (junta 4), conforme mostrado na Figura 3.1 [Epson, 2016].

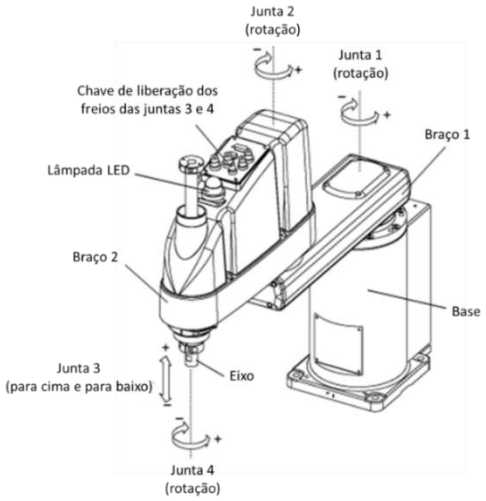


Figura 3.1 - Elementos do manipulador Epson SCARA.

O sistema de referência do robô corresponde a três eixos ortogonais X, Y e Z, e um eixo de rotação U, associado ao giro do efetuador e referente ao 4º GDL do manipulador. A representação do sistema de referência do manipulador é apresentado na Figura 3.2 [Epson, 2016]. O controlador do manipulador robótico é conectado ao computador através de uma conexão *ethernet*, que faz a comunicação entre o software de desenvolvimento EPSON RC+ 5.0 e o controlador C180.

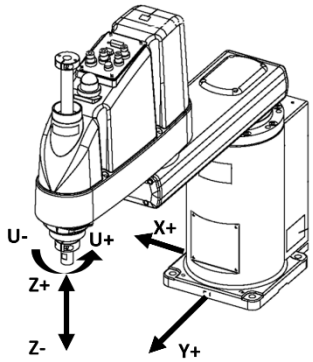


Figura 3.2 - Sistema de referência do manipulador Epson.

3.1.2 Equipamento de captura de imagem

Para a captura das imagens utiliza-se uma câmera de tecnologia CMOS. Devido as opções disponíveis no Laboratório de Automação e Robótica, optou-se por uma câmera do tipo *webcam*, da marca Logitech, modelo C920, com resolução *Full HD* 1080p e interface com o computador através de conexão USB (*Universal Serial* Bus), uma câmera compacta e leve, que atende as necessidades desse projeto. Dois fatores são importantes na escolha dessa câmera: a leveza da câmera, 162 g, câmera adequada para ser acoplada ao braço do robô SCARA, sem interferir na capacidade de carga do próprio robô; e a conexão USB com o computador, que elimina a necessidade de utilização de uma placa de captura de vídeo. O modelo de câmera utilizada é ilustrado na Figura 3.3.

Figura 3.3 - Câmera Logitech C920

A câmera possui algumas configurações que podem ser ajustadas e vários testes foram realizados para estabelecer a melhor configuração a ser utilizada no trabalho. As configurações utilizadas estão na Tabela 3.1.

Tabela 3.1 - Configurações da câmera

Parâmetro	**Configuração utilizada**	**Configuração disponível**
Resolução	1920x1080 pixels	várias
Modo de exposição	Manual	Manual ou Auto
Exposição	-8	-11 a -2
Ganho	125	0 a 255
Modo de balanço de branco	Manual	Manual ou Auto
Balanço de branco	5000	2000 a 6500
Foco	5	0 a 250

A câmera foi instalada em um suporte que foi acoplado ao braço do manipulador SCARA, próximo ao efetuador e solidária ao movimento do mesmo. As imagens capturadas para o treinamento da rede neural e para realizar o teste de reconhecimento de posições e orientações das peças foram capturadas com o TCP (*Tool Center Point*) do manipulador posicionado em um mesmo ponto fixo, de acordo com o sistema de referência do manipulador.

3.1.3 Hardware de processamento

Para o sistema de processamento de imagens e interface com o controlador do robô, o hardware adotado está baseado em um computador (*desktop*) com a configuração:

- Processador Intel Core i5 3.2GHz
- 4Gb de memória RAM
- Placa de vídeo integrada Intel HD *Graphics* 4600 com 256Mb de memória dedicada.
- Sistema Operacional *Windows 7 Professional 64 bits - Service Pack 1*

3.1.4 Software para implementação do sistema

Para a implementação do sistema desenvolvido utiliza-se o software MATLAB® (*MATrix LABoratory*), que é um software dedicado a fazer cálculos com matrizes e seus comandos são muito próximos da forma como se escreve as expressões algébricas, tornando seu uso simples e intuitivo. Todas as variáveis são dimensionadas automaticamente ao serem

atribuídas pela primeira vez e permanecem na memória de trabalho (*workspace*) até serem apagadas.

O MATLAB® é considerado um software e também uma linguagem de programação, e foi desenvolvido incialmente para computação numérica, originalmente desenvolvido pelo professor Cleve Moler nos anos 1970 para proporcionar fácil acesso a bibliotecas de álgebra linear. Em 1984 a empresa MathWorks foi fundada para continuar o desenvolvimento do produto [Paluszek e Thomas, 2015].

O MATLAB® é uma ferramenta de análise de dados, prototipagem e visualização com suporte para matrizes e operações matriciais, excelente capacidade gráfica e linguagem de programação de alto nível. O MATLAB® se tornou muito popular entre engenheiros, cientistas e pesquisadores, tanto na indústria como nas universidades, devido a vários fatores, entre eles a rica variedade de conjuntos de funções especiais – encapsuladas nas chamadas *toolboxes* – para muitas áreas de aplicação, como redes neurais, finanças, processamento de imagens, etc. O tipo de dados básico do MATLAB® são as matrizes, ou seja, de alguma maneira todos os dados inseridos no MATLAB® são matrizes ou algo derivado delas, por exemplo, números simples são considerados matrizes 1x1 [Marques, 2011].

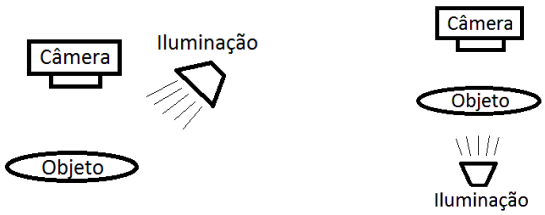
Segundo Matsumoto, 2006, programar em MATLAB® é mais simples, rápido e eficiente do que em qualquer outra linguagem de programação convencional como C/C++, VBA ou Fortran.

Neste trabalho são utilizadas algumas bibliotecas de rotinas do MATLAB®, denominadas de *toolbox.* A *toolbox* de Redes Neurais foi utilizada para treinamento da rede neural no reconhecimento da orientação das peças, e esse processo será melhor detalhado em seções posteriores.

3.2 Iluminação da área de trabalho

As técnicas de iluminação normalmente utilizadas para captura de imagens que serão processadas são a iluminação direta e iluminação traseira. Na iluminação direta a fonte de energia está sobre a área de trabalho na mesma direção da câmera, e pode ser iluminação do próprio ambiente, arranjo de lâmpadas posicionadas uniformemente sobre a área, ou algum arranjo específico que proporcione a melhor situação para captura das imagens e posterior processamento (Figura 3.4(a) e Figura 3.5(a)). Na iluminação traseira a fonte de energia está

em baixo da área de trabalho, e através de uma superfície transparente ela ilumina as peças no sentido oposto a câmera, gerando na imagem uma sombra onde estão as peças e um fundo iluminado onde a luz atravessa a superfície transparente sem obstáculos (Figura 3.4(b) e Figura 3.5(b)).



(a) (b)

Figura 3.4 - Técnicas de iluminação: iluminação direta (a) e iluminação traseira (b).

(a) (b)

Figura 3.5 - Exemplo de imagens com iluminação direta (a) e iluminação traseira (b).

Neste trabalho utiliza-se a técnica de iluminação direta, ou seja, somente a iluminação ambiente do laboratório. Durante os testes constatou-se que a iluminação existente era suficiente para proporcionar resultados satisfatórios na execução da tarefa de segmentação. A Figura 3.6 apresenta as condições de iluminação do local de testes.

Figura 3.6 - Iluminação do local dos testes

3.3 Configuração física dos testes

Os testes foram realizados na bancada onde está instalado o manipulador Epson SCARA G6, no Laboratório de Automação e Robótica da UFRGS.

Na área de trabalho determinada para captura das peças foi utilizado um fundo preto fosco sobre um suporte, para melhorar o contraste entre as peças, que são metálicas, e o plano de fundo nas imagens. A câmera que realiza a captura das imagens está instalada em um suporte fixado no braço do manipulador, próxima ao efetuador, e conectada à um computador através de uma conexão USB. Nesse computador está instalado o software utilizado para desenvolvimento do método, que realiza a captura e processamento das imagens, e determina a posição e orientação das peças, e também o software que se comunica com o controlador do manipulador. A câmera está ajustada paralelamente ao plano onde as peças estão posicionadas, para obter uma imagem de topo das peças, com os eixos c (coluna) e r (linha) da imagem paralelos aos eixos X e Y do manipulador, respectivamente. Conforme mencionado na seção 2.1.2, o eixo r, correspondente as linhas de pixels da imagem, possui direção de cima para baixo, diferente do eixo Y do manipulador, que tem direção de baixo para cima, sendo necessário realizar a inversão dos valores durante o cálculo da posição do centro de massa da peça em milímetro. A Figura 3.7 apresenta a configuração física da bancada de teste.

Figura 3.7 - Bancada de teste

3.4 Sistema desenvolvido

O sistema desenvolvido tem como foco a determinação da posição e orientação de vários objetos distribuídos sobre a área de trabalho (campo de visão da câmera), e subsequente

acionamento de um manipulador robótico para realizar a coleta dos objetos e transporte para um local definido. O sistema tal como projetado e implementado pode ter aplicações em sistemas de embalagem, ordenando certo número de objetos em um contentor, bem como na função de classificação e ordenamento de componentes por características geométricas ou mesmo no controle de características de qualidade. A lógica de implementação do sistema e seu funcionamento é ilustrado no fluxograma da Figura 3.8, seguido de uma descrição de seus módulos.

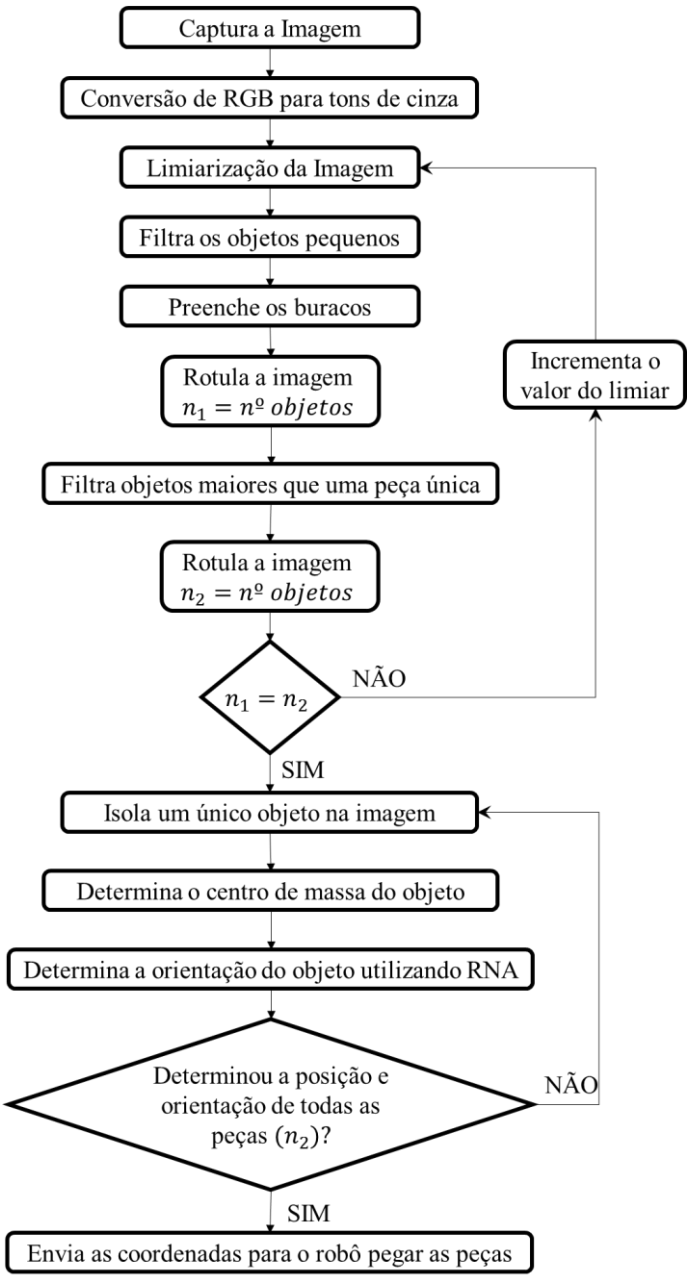


Figura 3.8 - Fluxograma do método desenvolvido

Para efeito de testes, as peças escolhidas são eixos metálicos provenientes de trabalhos práticos realizados por alunos da graduação no laboratório de usinagem, e embora não sejam perfeitamente iguais, os testes mostraram que o sistema desenvolvido é robusto na identificação da posição e orientação dessas peças. A Figura 3.9 mostra 5 peças distribuídas aleatoriamente sobre a área de trabalho, peças semelhantes mas não idênticas.

Figura 3.9 - Peças espalhadas aleatoriamente

3.4.1 Captura da imagem

A captura da imagem é realizada utilizando a câmera Logitech C920, por meio de software específico utilizado na implementação do sistema. A câmera é conectada ao computador por uma porta USB, e através de comandos via software é inicializada e configurada com os parâmetros definidos para realizar a captura das imagens. Após a captura, a imagem é salva na área de trabalho (*workspace*) do software MATLAB®, sendo então convertida de RGB para tons de cinza utilizando a função *rgb2gray*. Essa conversão de RGB para tons de cinza é uma soma ponderada dos valores do pixel nos canais R, G e B, segundo a Equação 3.1.

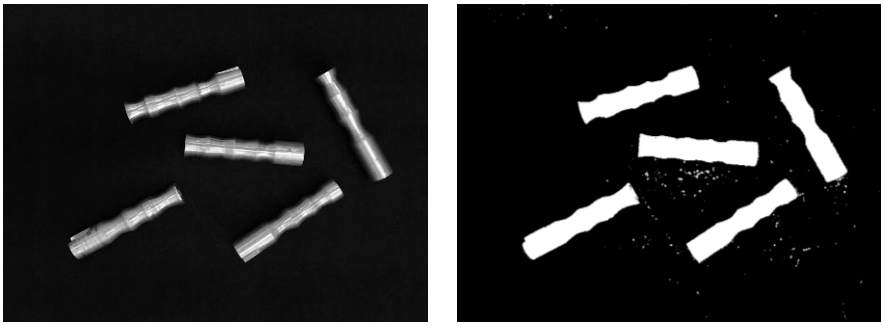
$$Valor\ do\ pixel\ cinza = 0{,}2989 \times R + 0{,}5870 \times G + 0{,}1140 \times B \qquad (3.1)$$

As imagens são capturadas com 8 bits, portanto existem 256 tons de cinza para cada pixel que forma a imagem, conforme explicado na seção 2.1. No módulo de captura o procedimento é implementado pelas seguintes instruções de código:

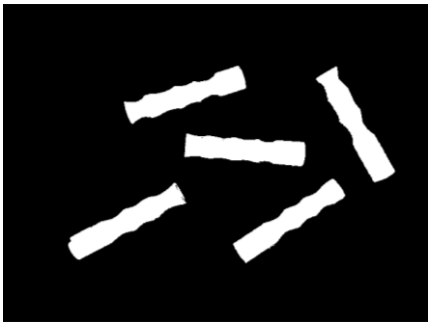
```
%atribui a webcam a variável
webcam = webcam(2);

%Configuração das propriedades da webcam
webcam.Resolution='1920x1080';
webcam.ExposureMode= 'manual';
webcam.Gain= 125;
webcam.WhiteBalanceMode= 'manual';
webcam.WhiteBalance= 5000;
webcam.Exposure= -2;
webcam.Focus= 5;

%captura foto
imagem=snapshot(webcam);
imwrite(teste,'RGB_imagem.jpeg','jpg');  %salva a imagem RGB da câmera no
disco
imagem_gray=rgb2gray(imagem);            %converte a foto para escala de
cinza
imwrite(imagem_gray,'imagem_gray.jpeg','jpg'); %salva a imagem cinza da
câmera
```

3.4.2 Pré-processamento e segmentação das imagens

Nessa etapa, a imagem capturada e em tons de cinza é limiarizada, utilizando um valor de limiar que gera o melhor resultado para a segmentação entre peças e plano de fundo (*background*). Configurada a câmera, são capturadas várias imagens utilizando valores de exposição (*Exposure*) entre -7 e -1, e aplicados às imagens limiares entre 0 e 255, para identificar qual o melhor valor para as condições de iluminação presentes durante o teste, identificando-se que para as condições presentes nos testes, o valor limiar de 20 era o que apresentava os melhores resultados. A Figura 3.10 apresenta a imagem original em tons de cinza e a imagem limiarizada utilizando valor limiar 20, e nota-se que além das peças identifica-se ruído na imagem limiarizada. Para remover o ruído utiliza-se uma função do MATLAB® chamada *bwareaopen*, que remove objetos com tamanho em pixels menor que um valor pré-determinado. Durante os testes de limiarização das imagens, identificou-se que para as configurações e arranjo físico adotados nesse trabalho, o tamanho de uma única peça apresenta valores entre 24000 e 36000 pixels, então considerou-se como ruído qualquer objeto com tamanho menor que 24000 pixels (Figura 3.10 (c)). Para preencher qualquer falha presente nas peças identificadas após a remoção do ruído, utiliza-se o comando *imfill*, que realiza operações de reconstrução morfológica (ver seção 2.3) e preenche as falhas presentes na peça, tornando-a uma peça sólida.



(a) (b)



(c)

Figura 3.10 - Imagem em tons de cinza (a), limiarizada com valor 20 (b) e após remoção do ruído (c).

Após o processamento obtêm-se uma imagem onde aparecem somente as peças (parte branca nas imagens) e o plano de fundo (parte preta da imagem). A partir dessa imagem as peças identificadas são rotuladas, para posteriormente isolar cada uma delas e verificar a sua orientação individualmente. A rotulagem é realizada pela função *bwlabel* do MATLAB®, que identifica os objetos individualmente e atribui um rótulo numérico único a cada um deles.

Adicionalmente faz-se uma verificação para identificar se os objetos reconhecidos na imagem representam uma única peça, ou podem agrupar mais de uma peça, que por estarem muito próximas umas das outras ou se tocando geraram um objeto maior na imagem. Para isso, após a rotulagem das peças é aplicada à imagem outro filtro do MATLAB® chamado *bwareafilt*, que mantém na imagem somente objetos com tamanho compreendido ente um limite inferior e superior, definidos de acordo com a aplicação específica. Após esse filtro a imagem é novamente rotulada, e se o número de peças identificadas antes do filtro for igual ao número de peças identificadas após o filtro, a identificação das peças está correta e parte-se para identificação da posição e orientação das mesmas. Quando o número de peças identificadas

antes do filtro for diferente do número de peças identificadas depois do filtro, significa que existem peças próximas ou se tocando e formando um objeto maior que o limite superior definido, sendo então aplicado um incremento no valor do limiar através de um *loop* de verificação, limiarizando novamente a imagem e repetindo o processo até que o número de peças identificadas antes e após o filtro sejam iguais, resultando na segmentação correta das peças na imagem.

3.4.3 Determinação da posição e orientação das peças

Após a rotulagem de todos objetos presentes na imagem, cada objeto é isolado em uma nova imagem e determinadas a respectiva posição do centro de massa e orientação. Esse processo de determinação das coordenadas é realizado para cada objeto, até que todos sejam identificados na imagem e tenham suas coordenadas definidas.

3.4.3.1 Determinação do centro de massa da peça

Para a definição das coordenadas de posição do centro de massa é utilizada a propriedade *centroid* da função *regionprops* do MATLAB®, que identifica o objeto formado pelo conjunto de pixels interconectados e calcula seu centro de massa. Essas coordenadas são relativas ao sistema de referência da imagem [c r] (ver seção 2.1.2). A Figura 3.11 mostra a imagem com 5 peças identificadas e seus respectivos centros de massa.

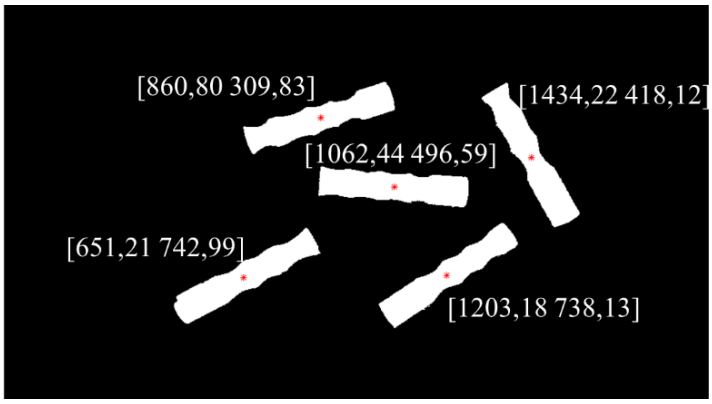


Figura 3.11 - Coordenadas dos centros de massa das peças

3.4.3.2 Rede Neural para determinação da orientação do objeto

Conforme exposto na seção 2.5, redes neurais artificiais são importantes ferramentas utilizadas em diversas áreas, e umas delas é o reconhecimento de padrões. A escolha da utilização de redes neurais para determinar a orientação das peças se deve ao fato de ser uma área de estudo que está se desenvolvendo muito nos dias atuais, e que oferece muitas possibilidades de utilização. Também foi considerado que o software utilizado para implementação do método possui um *toolbox* destinado a estruturar o treino de redes neurais, possibilitando executar essa tarefa de forma precisa e eficiente.

Para o presente trabalho uma rede neural foi treinada utilizando 9 peças semelhantes como modelo de padrão a ser reconhecido. A partir das imagens individuais de cada peça foi realizada a limiarização manual das mesmas, a fim de obter uma imagem com o padrão da peça bem definido e sem ruídos (Figura 3.12). Como o objetivo da rede neural é reconhecer a orientação da peça, foi determinado que o conjunto de amostras que seria utilizado para treinamento da RNA seria formado por imagens das 9 peças com alteração na orientação a cada 3° (1°, 4°, 7°, ... , 358°). Esse valor de passo da orientação foi definido em 3° devido principalmente a dois fatores: para que o banco de amostras gerado não fosse muito grande, visto que 9 peças orientadas a cada 3° gera um banco de amostra de 1080 imagens; e que a orientação do efetuador dentro desse intervalo de ± 3° não interfere na operação de coleta da peça dadas as características do efetuador.

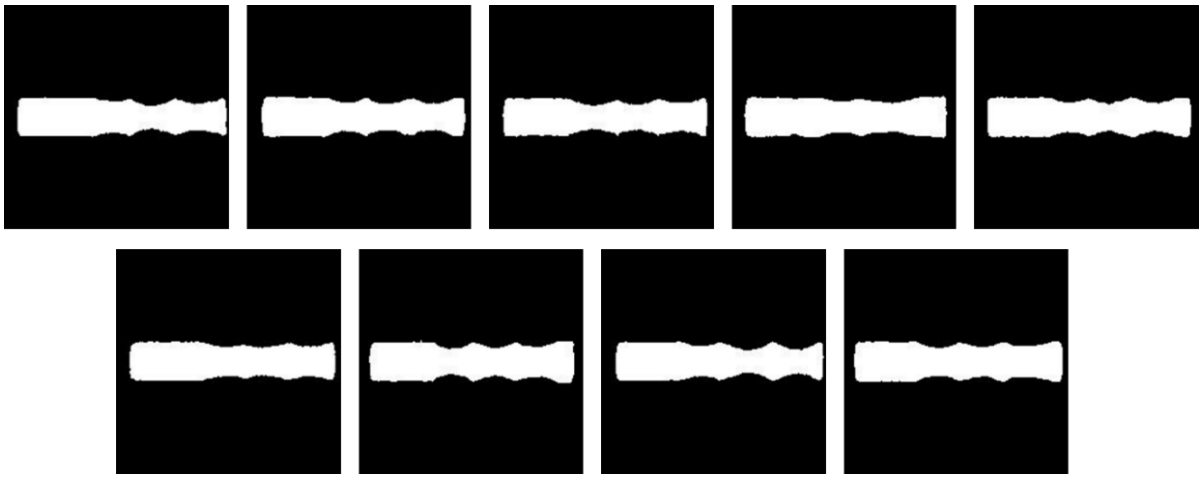


Figura 3.12 - Imagens das peças padrão para treino da RNA

A geração de um banco de amostras com muitas imagens para treino da RNA gera uma rede neural muito grande, o que prejudica muito a velocidade de reconhecimento do padrão. Considera-se também que esse trabalho é realizado utilizando computadores de uso geral, e não centrais de processamento dedicadas a somente processar o algoritmo e realizar a tarefa de reconhecimento das coordenadas das peças, o que justifica a utilização de um número menor de imagens.

As imagens utilizadas como amostras são recortes quadrados centrados na peça, retirados das imagens capturadas pela câmera, que foram redimensionados para resolução 200 x 200 pixels. A partir das 1080 imagens geradas – 9 amostras em cada orientação, sendo as orientações definidas a cada 3° – cada imagem foi redefinida para um vetor coluna e os 1080 vetores formaram a matriz de entrada da rede neural chamada *input*. Foi definida também uma matriz *target,* que corresponde a relação entre os dados da matriz *input* e a correspondente orientação da peça.

Para o treinamento da rede neural foi utilizado o *toolbox* chamado *Neural Net Pattern Recognition*, que arquiteta uma rede neural de duas camadas com alimentação para frente, sendo a camada escondida com o número configurável de neurônios e função de ativação do tipo sigmoide, e a camada de saída com 120 neurônios e função de ativação do tipo softmax, onde as saídas correspondem as orientações da peça a cada 3°. Para a configuração de treino da rede neural foi utilizado 80% das amostras para o treino, 10% das amostras para validação do treino e 10% das amostras para testar a rede neural, parâmetros configurados no *toolbox*.

O treino das redes neurais foi realizado utilizando 40, 80 e 120 neurônios na camada escondida, para determinar a melhor configuração a ser utilizada, e a rede neural obtida após o treinamento é utilizada para reconhecer a orientação das peças encontradas na imagem segmentada. Os números de neurônios na camada escondida foram escolhidos por serem múltiplos do número de orientações para qual a rede neural estava sendo treinada. Durante o treino da rede neural buscou-se como resultado uma porcentagem de erro igual a 0%, que significa que a parte das amostras utilizadas para testar a rede foram todas classificadas corretamente, um valor de erro diferente de zero significa que as amostras de teste estão sendo classificadas incorretamente, sendo necessário alterar os parâmetros da rede para melhorar o treino e diminuir o erro a zero.

Quando a rede neural é aplicada à imagem da peça segmentada, como resultado são obtidos valores na camada de saída da rede neural em forma de um vetor com 120 posições,

onde cada posição representa uma das orientações definidas no treino, que variam entre 1° e 358°, com intervalos de 3°. Nesse vetor estão os valores de probabilidade de a peça estar em uma das 120 orientações, e determina-se a orientação da peça como sendo a posição no vetor que possui o maior valor.

A Tabela 3.2 compara os resultados obtidos na determinação da orientação das peças identificadas na Figura 3.10(a), utilizando redes neurais treinadas com 40, 80 e 120 neurônios na camada escondida. As orientações das peças, aqui chamadas de reais, foram obtidas utilizando a propriedade *orientation* da função *regionprops* do MATLAB® (Figura 3.13), e todas as orientações são dadas em graus.

Tabela 3.2 - Comparação utilizando 40, 80 e 120 neurônios na camada escondida da RNA

Peça	Orientação real		40 neurônios			80 neurônios			120 neurônios		
1	28,0	orientação	28	91	31	28	25	205	28	25	31
		probabilidade	99,77%	0,03%	0,02%	99,99%	1,46E-05	1,43E-05	99,99%	0,01%	6,35E-06
2	197,5	orientação	199	205	196	199	196	106	199	196	16
		probabilidade	30,54%	13,84%	6,75%	58,07%	37,31%	1,13%	98,14%	1,70%	0,05%
3	175,6	orientação	178	175	280	175	178	355	175	178	355
		probabilidade	91,14%	4,62%	0,95%	93,87%	3,80%	1,34%	80,46%	15,86%	3,49%
4	33,6	orientação	34	37	67	34	31	109	34	31	37
		probabilidade	99,90%	0,03%	0,03%	100,00%	2,30E-05	4,32E-06	99,98%	0,01%	1,72E-05
5	119,3	orientação	118	211	151	118	121	151	118	121	127
		probabilidade	60,87%	20,12%	3,58%	86,48%	13,50%	2,69E-05	99,55%	0,41%	0,02%

A tabela apresenta as três primeiras orientações encontradas pela RNA com maior probabilidade de ser a orientação da peça. Nota-se que conforme aumenta o número de neurônios na camada escondida, a probabilidade melhora para quase todas as peças, e que independentemente do número de neurônios, todas as RNA encontraram como ângulo mais provável o mesmo resultado. Por exemplo para peça rotulada como 5, todas as RNA encontraram o valor de 118° para orientação da peça como primeira resposta, cuja orientação real é 119,3°, e a probabilidade passou de 60,87% com a rede neural de 40 neurônios para 99,55% para a rede neural de 120 neurônios. A partir dos resultados foi adotada a configuração com 120 neurônios na camada escondida.

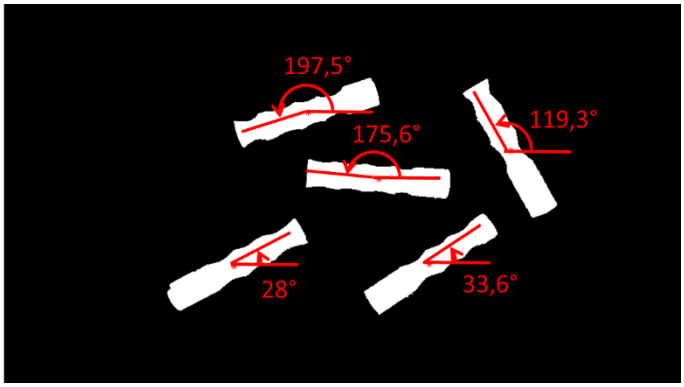


Figura 3.13 - Orientação real das peças

Também foram realizados testes treinando as redes neurais utilizando imagens das peças com resolução menor, 100 x 100 e 50 x 50 pixels, com o objetivo de tornar a identificação da orientação mais rápida. Para isso foi escolhida a rede neural com 120 neurônios na camada escondida, que apresentou melhor resultado no teste anterior. A Tabela 3.3 mostra os valores encontrados durante o teste.

Tabela 3.3 - Comparação utilizando resoluções diferentes no treino da rede neural

Peça	Orientação real		120 neurônios 50x50 pixels			120 neurônios 100x100 pixels			120 neurônios 200x200 pixels		
1	28,03	orientação	28	31	25	28	31	25	28	25	31
		probabilidade	82,47%	17,02%	0,32%	100,00%	5,73E-06	5,72E-06	99,99%	0,01%	6,35E-06
2	197,5	orientação	199	196	193	199	196	19	199	196	16
		probabilidade	65,57%	29,79%	2,03%	62,21%	34,94%	2,09%	98,14%	1,70%	0,05%
3	175,6	orientação	178	175	355	175	178	172	175	178	355
		probabilidade	51,98%	36,77%	9,31%	99,83%	0,16%	4,72E-05	80,46%	15,86%	3,49%
4	33,6	orientação	34	31	37	34	31	37	34	31	37
		probabilidade	99,44%	0,24%	0,09%	99,99%	0,01%	2,90E-05	99,98%	0,01%	1,72E-05
5	119,3	orientação	121	118	115	118	121	124	118	121	127
		probabilidade	92,43%	6,67%	0,22%	61,00%	38,87%	0,04%	99,55%	0,41%	0,02%

Neste segundo teste percebe-se que a rede neural treinada com imagens de 200 x 200 pixels apresentou melhores resultados, obtendo como resposta uma probabilidade maior em quase todos os casos, e os valores encontrados para orientação próximos a orientação real. Foi escolhida a resolução de 200 x 200 pixels para ser utilizada neste trabalho.

3.4.4 Envio das coordenadas para o controlador do manipulador

Após a captura e processamento da imagem, e determinação das coordenadas das peças, as informações de posição e orientação devem ser enviadas ao controlador do manipulador robótico, para que o mesmo colete as peças e coloque em outro local, definido previamente.

Para envio das informações é necessário estabelecer a comunicação entre o software de implementação do sistema e a interface que conecta ao controlador do manipulador, que estão instalados no mesmo computador. Essa comunicação é realizada através de *sockets* TCP/IP, que conforme explicado na seção 2.6, utiliza protocolos de comunicação que garantem o envio e recebimento dos dados de forma integra e ordenada.

O software de implementação é definido como servidor e nele é configurado um *socket* TCP/IP utilizando o endereço IP padrão 0.0.0.0 e a porta 5000. O software do controlador do manipulador é configurado como cliente e cria-se outro *socket* TCP/IP, com endereço IP da placa de rede *ethernet* que está conectada ao controlador (192.168.0.110) e a mesma porta configurada no servidor, porta 5000. A Figura 3.14 mostra o esquema de comunicação entre os softwares e o robô.

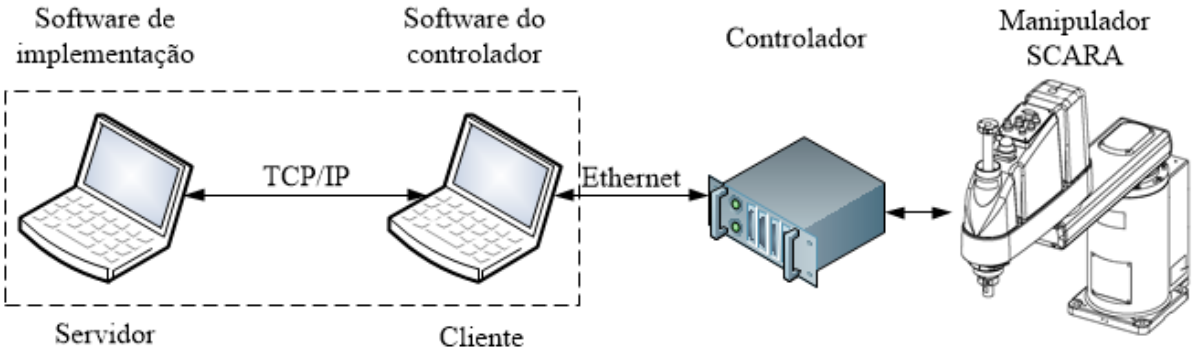


Figura 3.14 - Comunicação entre os softwares e o robô

A comunicação no sistema ocorre quando se executa ao mesmo tempo o algoritmo no MATLAB® e o código programado no software do controlador, assim a conexão é estabelecida e os valores da posição e orientação das peças são enviados ao controlador do manipulador para realizar a coleta das peças.

3.4.4.1 Cálculo das coordenadas referenciadas ao manipulador

Uma vez determinadas as coordenadas das peças na imagem e envio das mesmas para o software do controlador, é necessária a conversão dos valores encontrados para os centros de massa, que estão em pixels, para milímetros. Para o manipulador realizar a coleta das peças, os valores enviados para o controlador devem estar em milímetros e com as coordenadas relacionadas ao sistema de referência adotado pelo manipulador.

Para realizar a conversão das medidas de pixel para milímetro foram mensuradas axialmente as 9 peças da Figura 3.12, utilizando um paquímetro Mitutoyo com resolução de 0,02 mm, e foram relacionados os valores encontrados com o tamanho das respectivas peças obtidos nas imagens em pixels, conforme indicados na Tabela 3.4.

Tabela 3.4 - Relação pixels x milímetros

Peça	Tamanho		Relação Pixels/mm
	mm	pixels	
1	82,46	445,88	5,407
2	80,06	433,85	5,419
3	80,02	435,06	5,437
4	79,8	432,24	5,417
5	80	433,01	5,413
6	80,58	436,76	5,420
7	80,36	436,96	5,437
8	81,48	442,61	5,432
9	82,16	444,66	5,412

A partir desses resultados é calculada a média aritmética dessa relação, que tem o valor de $r_p = 5{,}421\ pixels/mm$.

Os valores para o centro de massa das peças são encontrados com base no sistema de referência da imagem (ver seção 2.1.2), onde o centro focal da câmera corresponde ao centro da imagem, porém o efetuador do manipulador está deslocado em relação a esse centro, sendo necessário corrigir essa excentricidade. Por meio de testes se determinou esses valores como sendo $x = -50\ mm$ e $y = -48\ mm$, do efetuador em relação ao centro focal da câmera. A Figura 3.15 ilustra os sistemas de referência da câmera e do manipulador, e a Figura 3.15 e Figura 3.16 mostram a excentricidade entre a câmera e o eixo do efetuador.

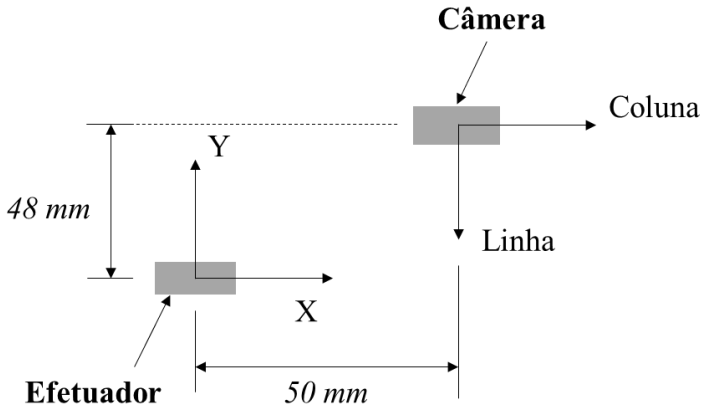


Figura 3.15 - Sistemas de referências e excentricidade entre a câmera e o efetuador

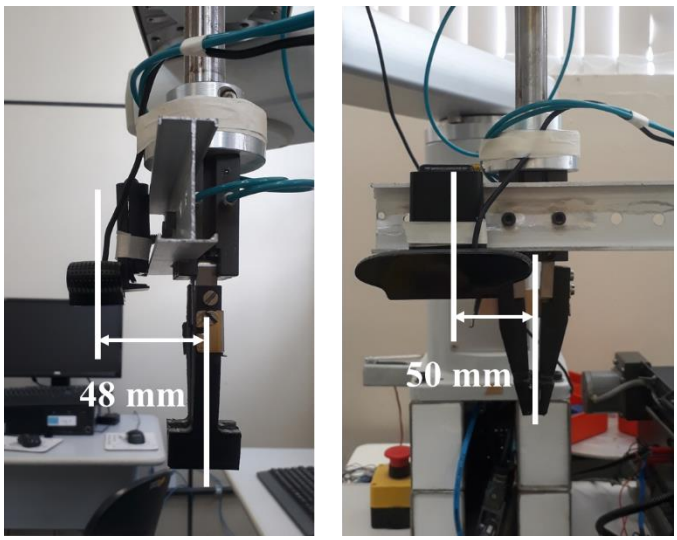


Figura 3.16 – Imagens indicativas para excentricidade entre câmera e eixo do efetuador.

Para calcular a posição do centro de massa das peças em milímetros em relação ao sistema de referência do manipulador, além da relação pixels/mm e da excentricidade da câmera em relação ao efetuador, é necessário considerar a posição da câmera quando a imagem é capturada, valores considerados no cálculo e apresentados nas próximas seções.

4 TESTES E AVALIAÇÃO DO SISTEMA

Para avaliar o sistema de visão computacional proposto foram realizados testes com diferentes números de peças, dispostas aleatoriamente na área de trabalho, para diferentes valores de velocidade e aceleração/desaceleração do manipulador, que representam 12,5% e 50% dos valores máximos de operação. Em cada teste avaliou-se o tempo de execução, foram registradas as coordenadas de cada peça (posição e orientação), e a probabilidade da orientação encontrada ser uma das poses pertencente ao banco de dados utilizado para treino da rede neural, também foram determinados os valores da orientação real das peças utilizando a propriedade *orientation* do MATLAB®.

A imagem é capturada com o manipulador posicionado em um ponto fixo, definido de acordo com o sistema de referência do manipulador, nas coordenadas $X = -30\ mm, Y = 375\ mm, Z = -80\ mm$ *e* $U = -68{,}5°$.

O fluxograma da Figura 4.1 apresenta as etapas do teste, realizado para 3 amostras de peças.

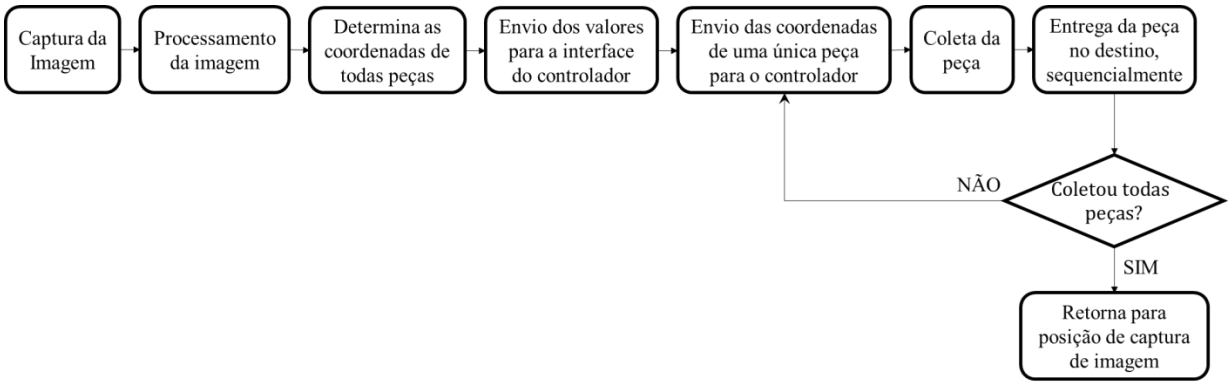
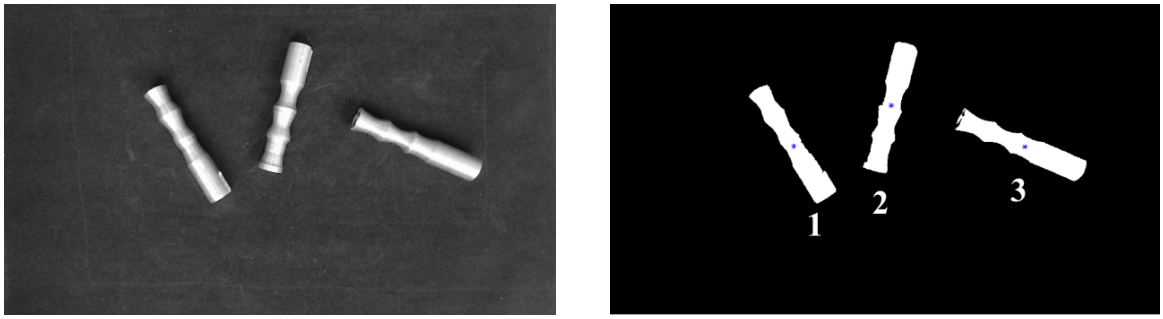


Figura 4.1 - Fluxograma das etapas do teste

4.1 Teste 1 – Amostra de 3 peças

Neste teste foram utilizadas 3 peças dispostas aleatoriamente sobre a área de trabalho. A Figura 4.2 apresenta a imagem capturada (a) e a imagem após o processamento (b), com os centros de massa e a rotulagem das peças identificadas.



(a) (b)

Figura 4.2 - Imagem das 3 peças

Foram realizados 3 testes com 12,5% da velocidade (v) e aceleração/desaceleração (a) máxima do manipulador, que correspondem a $v = 250\ mm/s$ e $a = 312{,}5\ mm/s^2$, e 3 testes com 50% dos valores máximos, que correspondem a $v = 1000\ mm/s$ e $a = 1250\ mm/s^2$.

Considera-se um ciclo desde o comando de início nos softwares de desenvolvimento e do controlador do robô, até o manipulador retornar à posição de captura da imagem, após coletar todas as peças.

Os dados coletados durante os testes são apresentados na Tabela 4.1 e Tabela 4.2.

Tabela 4.1 - Tempo dos testes com 3 peças

Parâmetros do controlador	Teste	Tempo (s)
$v = 250\ mm/s$ $a = 312{,}5\ mm/s^2$	1	34,46
	2	34,12
	3	34,36
$v = 1000\ mm/s$ $a = 1250\ mm/s^2$	1	23,99
	2	23,76
	3	23,74

Tabela 4.2 – Dados da posição e orientação das 3 peças

Peça	Posição *c* na coluna (pixels)	Posição *r* na linha (pixels)	Orientação (graus)	Probabilidade (%)	Orient. real (graus)
1	646,33	496,66	124	96,17	122,59
2	983,31	355,13	256	99,99	255,68
3	1444,03	497,54	157	99,81	156,66

Nos testes com velocidade de $250\ mm/s$ e aceleração/desaceleração de $312{,}5\ mm/s^2$ o tempo médio de um ciclo foi de $34{,}31\ s$, e nos testes utilizando velocidade de $1000\ mm/s$ e aceleração/desaceleração de $1250\ mm/s^2$ o tempo médio foi de $23{,}83\ s$.

As orientações encontradas pela rede neural utilizada no sistema são muito próximas

das orientações reais das peças, e o sistema obteve probabilidade acima de 96%, demonstrando a eficiência do método e a confiabilidade na determinação da orientação através de redes neurais.

As coordenadas encontradas são enviadas ao software que está conectado ao controlador do manipulador, onde são calculados os valores para a posição do centro de massa das peças, relativos ao sistema de referência do manipulador, e é corrigida a orientação de acordo com a posição "zero" do efetuador. As Equações 4.1, 4.2 e 4.3 demonstram os cálculos realizados para obter-se os valores de X, Y e U (orientação do efetuador) segundo o sistema de referência do controlador.

$$X = -30 + 50 + \frac{Posição\ c - 960}{5{,}4215} \quad (4.1)$$

$$Y = 375 + 48 + \frac{540 - Posição\ r}{5{,}4215} \quad (4.2)$$

$$U = orientação - 68{,}5 - 90 \quad (4.3)$$

Como as peças se encontram em um mesmo plano, a medida do eixo Z para coleta das peças é fixa e determinada experimentalmente como $Z = -260\ mm$. No ponto de descarga das peças a posição Z também é fixa e foi determinada como $Z = -171\ mm$. A Tabela 4.3 apresenta os valores calculados e enviados ao controlador.

Tabela 4.3 - Valores enviados ao controlador

Peça	Posição X (mm)	Posição Y (mm)	Posição Z (mm)	Orientação U (graus)
1	-37,85	430,99	-260	-34,50
2	24,30	457,10	-260	97,50
3	109,28	430,83	-260	-1,50

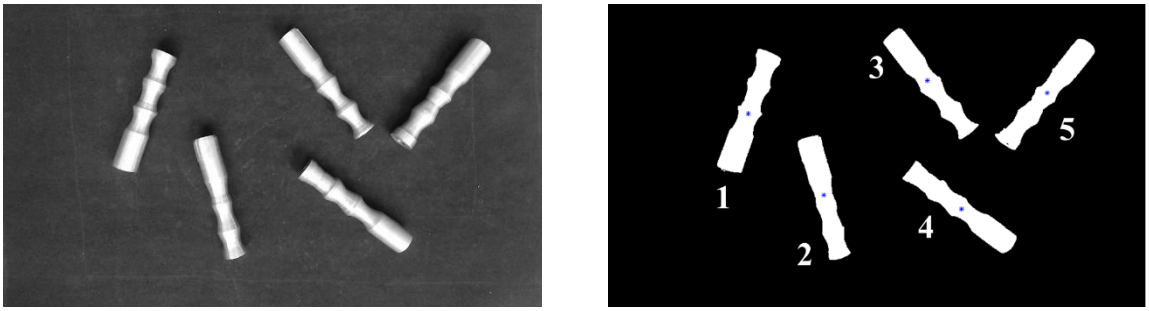
Os dados são enviados ao controlador, que executa a tarefa de pegar cada uma das peças e entregar no ponto de descarga tomado como referência inicial ($X = -275\ mm, Z = -171\ mm\ e\ U = 21{,}5°$), uma ao lado da outra em intervalos de $40\ mm$ a partir de $Y = 200\ mm$. A Figura 4.3 mostra as peças na área de trabalho (a), o manipulador movendo-as para o destino (b) e as peças uma ao lado da outra no destino final (c).

(a) (b) (c)

Figura 4.3 - Coleta das peças durante o teste

4.2 Teste 2 – Amostra de 5 peças

O mesmo teste foi aplicado para 5 peças dispostas aleatoriamente sobre a área de trabalho. A Figura 4.4 apresenta a imagem capturada (a) e a imagem após o processamento (b), com os centros de massa e a rotulagem das peças identificadas.



(a) (b)

Figura 4.4 - Imagem das 5 peças

Seguindo os mesmos parâmetros utilizados no teste anterior, para velocidade e aceleração do manipulador, foram obtidos os valores apresentados na Tabela 4.4 e Tabela 4.5.

Tabela 4.4 - Tempo dos testes com 5 peças

Parâmetros do controlador	Teste	Tempo (s)
$v = 250\ mm/s$ $a = 312{,}5\ mm/s^2$	1	49,78
	2	49,60
	3	49,78
$v = 1000\ mm/s$ $a = 1250\ mm/s^2$	1	29,71
	2	29,40
	3	29,73

Tabela 4.5 – Dados da posição e orientação das 5 peças

Peça	Posição *c* na coluna (pixels)	Posição *r* na linha (pixels)	Orientação (graus)	Probabilidade (%)	Orient. real (graus)
1	508,138	393,569	70	94,32	70,38
2	772,894	678,901	286	98,91	284,80
3	1141,611	275,171	307	75,72	308,45
4	1260,908	728,476	142	99,96	142,99
5	1564,363	318,792	229	97,94	229,98

Nos testes com velocidade de $250\ mm/s$ e aceleração/desaceleração de $312{,}5\ mm/s^2$ o tempo médio de um ciclo foi de $49{,}72\ s$, e nos testes utilizando velocidade de $1000\ mm/s$ e aceleração/desaceleração de $1250\ mm/s^2$ o tempo médio foi de $29{,}61\ s$.

As orientações encontradas pela rede neural utilizada no sistema são muito próximas das orientações reais das peças, e o sistema obteve probabilidade acima de 94% para quase todas peças, exceto a peça 3 onde obteve-se a orientação de 307° com probabilidade de 75,72% e na segunda resposta obtida pela rede neural obteve-se 310° com probabilidade de 24,12%, isso ocorre devido ao valor do ângulo real (308,45°) estar próximo a metade do intervalo entre 307° e 310°. Apesar disso, verifica-se a eficiência do método e a confiabilidade na determinação da orientação através de redes neurais.

As coordenadas encontradas são enviadas ao software que está conectado ao controlador do manipulador, onde são calculados os valores para a posição do centro de massa das peças, relativos ao sistema de referência do manipulador, e é corrigida a orientação de acordo com a posição "zero" do efetuador. As Equações 4.1, 4.2 e 4.3 demonstram os cálculos realizados para obter-se os valores de X, Y e U, segundo o sistema de referência do controlador.

Como as peças se encontram em um mesmo plano, a medida do eixo Z para coleta das peças é fixa e foi determinada experimentalmente como $Z = -260\ mm$. No ponto de descarga das peças a posição Z também é fixa e foi determinada como $Z = -171\ mm$. A Tabela 4.6 apresenta os valores calculados e enviados ao controlador.

Tabela 4.6 - Valores enviados ao controlador

Peça	Posição X (mm)	Posição Y (mm)	Posição Z (mm)	Orientação U (graus)
1	-63,345	450,009	-260	-88,5
2	-14,511	397,380	-260	127,5
3	53,498	471,847	-260	148,5
4	75,502	388,236	-260	-16,5
5	131,473	463,801	-260	70,5

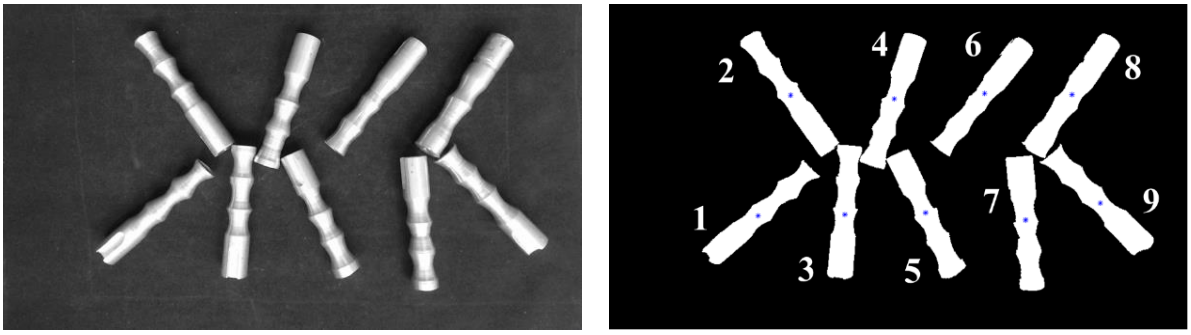
Os dados são enviados ao controlador, que executa a tarefa de pegar cada uma das peças e entregar no ponto de descarga ($X = -275\ mm, Z = -171\ mm\ e\ U = 21{,}5°$), uma ao lado da outra em intervalos de $40\ mm$ a partir de $Y = 200\ mm$. A Figura 4.5 mostra as peças na área de trabalho (a), o manipulador movendo-as para o destino (b) e as peças uma ao lado da outra no destino final (c).

(a) (b) (c)

Figura 4.5 - Coleta das peças durante o teste

4.3 Teste 3 – Amostra de 9 peças

O mesmo teste foi aplicado para 9 peças dispostas aleatoriamente sobre a área de trabalho. A Figura 4.6 apresenta a imagem capturada (a) e a imagem após o processamento (b), com os centros de massa e a rotulagem das peças identificadas.



(a) (b)

Figura 4.6 - Imagem das 9 peças

Foram utilizados os mesmos parâmetros dos dois testes anteriores e os dados coletados são apresentados na Tabela 4.7 e Tabela 4.8.

Tabela 4.7 - Dados de teste com 9 peças

Parâmetros do controlador	Teste	Tempo (s)
$v = 250\ mm/s$ $a = 312{,}5\ mm/s^2$	1	76,06
	2	76,08
	3	74,50
$v = 1000\ mm/s$ $a = 1250\ mm/s^2$	1	42,64
	2	42,72
	3	42,65

Tabela 4.8 – Dados da posição e orientação das 9 peças

Peça	Posição *c* na coluna (pixels)	Posição *r* na linha (pixels)	Orientação (graus)	Probabilidade (%)	Orient. real (graus)
1	501,395	700,449	43	85,19	41,33
2	609,603	305,281	124	87,42	125,72
3	784,072	697,448	85	99,96	85,53
4	947,409	318,660	250	99,37	251,22
5	1050,786	691,311	295	99,99	295,41
6	1246,634	299,884	232	99,92	231,10
7	1380,945	713,413	274	67,78	275,88
8	1533,378	302,296	235	99,94	234,81
9	1623,933	659,898	133	69,94	135,15

Nos testes com velocidade de $250\ mm/s$ e aceleração/desaceleração de $312{,}5\ mm/s^2$ o tempo médio de um ciclo foi de $75{,}55\ s$, e nos testes utilizando velocidade de $1000\ mm/s$ e aceleração/desaceleração de $1250\ mm/s^2$ o tempo médio foi de $42{,}67\ s$.

Assim como no teste com 5 peças, as orientações encontradas pela rede neural utilizada no sistema são muito próximas das orientações reais das peças, e o sistema obteve probabilidade acima de 99% para quase todas peças. Apesar da probabilidade menor na orientação de algumas peças, verifica-se que o valor para orientação real é próxima a metade do intervalo entre dois ângulos treinados na rede neural, e que a segunda resposta da rede neural é sempre o ângulo imediatamente inferior ou superior ao treinado, conforme apresentado na Tabela 4.9. Apesar desses valores menores na probabilidade da primeira resposta da rede neural, pode-se considerar que o método é eficiente e confiável na determinação da orientação através de redes neurais.

Tabela 4.9 -Valores da probabilidade para orientação encontrados pela rede neural

Peça	Orientação real (graus)	1ª resposta		2ª resposta	
		Orientação (graus)	Probabilidade (%)	Orientação (graus)	Probabilidade (%)
1	41,33	43	85,19	40	14,77
2	125,72	124	87,42	127	12,51
3	85,53	85	99,96	88	0,02
4	251,22	250	99,37	253	0,60
5	295,41	295	99,99	112	0
6	231,10	232	99,92	229	0,06
7	275,88	274	67,78	277	31,97
8	234,81	235	99,94	55	0,03
9	135,15	133	69,94	136	29,99

As coordenadas encontradas são enviadas ao software que está conectado ao controlador do manipulador, onde são calculados os valores para a posição do centro de massa das peças, relativos ao sistema de referência do manipulador, e é corrigida a orientação de acordo com a posição "zero" do efetuador. As Equações 4.1, 4.2 e 4.3 demonstram os cálculos realizados para obter-se os valores de X, Y e U (orientação do efetuador) segundo o sistema de referência do controlador.

Como as peças se encontram em um mesmo plano, a medida do eixo Z para coleta das peças é fixa e foi determinada experimentalmente como $Z = -260\ mm$. No ponto de descarga das peças a posição Z também é fixa para efeito de teste e foi determinada como $Z = -171\ mm$.

A

Tabela 4.10 apresenta os valores calculados e enviados ao controlador.

Tabela 4.10 - Valores enviados ao controlador

Peça	Posição X (mm)	Posição Y (mm)	Posição Z (mm)	Orientação U (graus)
1	-64,59	393,40	-260	-115,5
2	-44,63	466,29	-260	-34,5
3	-12,45	393,96	-260	-73,5
4	17,68	463,82	-260	91,5
5	36,74	395,09	-260	136,5
6	72,87	467,29	-260	73,5
7	97,64	391,01	-260	115,5
8	125,76	466,84	-260	76,5
9	142,46	400,88	-260	-25,5

Os dados são enviados ao controlador, que executa a tarefa de pegar cada uma das peças e distribuir a partir de um ponto inicial de descarga ($X = -275\ mm, Z = -171\ mm\ e\ U =$ 21,5°), uma ao lado da outra em intervalos de $40\ mm$ a partir de $Y = 200\ mm$. A Figura 4.7 mostra as peças na área de trabalho (a), o manipulador movendo-as para o destino (b) e as peças uma ao lado da outra no destino final (c).

(a) (b) (c)

Figura 4.7 - Coleta das peças durante o teste

5 CONCLUSÕES

A partir dos resultados apresentados nos testes, pode-se concluir que o sistema desenvolvido cumpre com o objetivo inicial proposto, considerando que o sistema é capaz de identificar objetos com geometria conhecida espalhados aleatoriamente na área de trabalho de um robô manipulador, determinar a posição e orientação de cada um e enviar essas informações para o controlador do manipulador robótico que realiza a coleta e transporte dos objetos para posições estabelecidas.

O objetivo foi alcançado, pois em todos os testes realizados as peças foram identificadas corretamente, seus centros de massa e orientação determinados, e o manipulador coletou e transportou corretamente todas as peças para o ponto de descarga determinado, sem nenhum erro. Esses resultados demonstram que o sistema apresentado é robusto e confiável para ser utilizado em condições análogas as utilizadas nos testes, porém pode ser empregado para coleta de outros modelos de peças ou em outras condições de iluminação, exigindo apenas o treino de uma nova rede neural e ajustes nos valores do limiar, para o correto funcionamento do sistema de visão computacional proposto.

5.1 Sugestões para trabalhos futuros

O sistema foi desenvolvido para realizar uma tarefa genérica de coleta e transporte de peças estáticas, sem uma aplicação baseada em algum processo industrial, porém como sugestões para trabalhos futuros pode-se listar:

- adequação do sistema para uma aplicação industrial específica;
- integração do sistema com uma esteira, para coleta de peças em movimento;
- adaptação do sistema para identificação de objetos diferentes, para ser utilizado também como classificador;
- aprimorar o sistema para identificar e determinar a posição e orientação de peças em 3 dimensões;

REFERÊNCIAS BIBLIOGRÁFICAS

Acharya, T. e Ray, A. K. **Image processing: Principles and applications**. Wiley, 2005.

Ansari, M. S. **Non-Linear Feedback Neural Networks - VLSI Implementations and Applications**. Springer, 2014.

Casasent, D., Talukder, A., Cox, W. e Chang, H.; Weber, D. Detection and Segmentation of Multiple Touching Product Inspection Items. **Optics in Agriculture, Forestry, and Biological Processing II**, vol. 2907, 1996.

Consultant, R. B. Random bin picking: has its time finally come? **Assembly Automation**, vol. 34, issue 3, p. 217-221, 2014.

Donahoo, M. J. e Calvert, K. L. **TCP/IP Sockets in C – Practical guide for programmers.** Elsevier, 2009.

Du, K. e Swamy, M.N.S. **Neural Networks and Statistical Learning**. Springer, 2014.

EPSON. **Epson Manipulator Manual SCARA robot G6 Series**. Rev. 13, 2016.

Fall, K. R. e Stevens, W. R. **TCP/IP Illustrated, Volume 1 – The protocols.** Addison-Wesley, 2012.

Girshick, R., Donahue, J., Darrell, T. e Malik, J. Region-Based Convolutional Networks for Accurate Object Detection and Segmentation. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, vol. 38, issue 1, p. 142-158, 2015a.

Girshick, R. Fast R-CNN**. IEEE International Conference on Computer Vision (ICCV)**, p. 1440 - 1448, 2015b.

Golnabi, H. e Asadpour, A. Design and application of industrial machine vision systems. **Robotics and Computer-Integrated Manufacturing**, vol. 23, issue 6, p. 630-637, 2007.

Gonzales, R. C. e Woods, R. E. **Processamento de imagens digitais**. 1. ed. Edgard Blücher, 2000.

Hagan, M. T., Demuth, H. B., Beale, M. H. e Jesus, O. **Neural Network Design.** 2nd edition. eBook, 1996.

Haykin, S. **Redes Neurais – Princípios e práticas.** Porto Alegre, Bookman, 2001.

He, X. e Xu, S. **Process Neural Networks – Theory and Applications**. Springer 2007.

Hu, M. Visual pattern recognition by moment invariants**. IRE Transactions on Information Theory**, vol. 8, no. 2, pp. 179-187,1962.

IFR – *International Federation of Robotics*. **Executive Summary World Robotics 2017 Industrial Robots.** https://ifr.org/free-downloads/, 2017. Acesso em: 28-09-2018.

International Organization for Standardization. **ISO 10218-1:2011 - Robots and robotic devices - Safety requirements for industrial robots - Part 1: Robots**, p. 43, 2011.

Ju, M., Choi, Y., Seo, J., Sa, J., Lee, S., Chung, Y. e Park, D. A Kinect-Based Segmentation of Touching-Pigs for Real-Time Monitoring. **Sensors**, vol. 18, issue 6, p. 1746, 2018.

Korath, J.M., Abbas, A. e Romagnoli, J.A. Separating touching and overlapping objects in particle images – A combined approach. **Chemical Engineering Transactions**, vol. 11, p. 167-172, 2007.

Kurose, J. F. e Ross, K. W. **Redes de computadores e a Internet: uma abordagem top-down**. Pearson education, 2013

Lowe, D. G. Distinctive Image Features from Scale-Invariant Keypoints. **International Journal of Computer Vision**, vol. 60, issue 2, p. 91–110, 2004.

Marques, O. **Practical image and video processing using MATLAB®**. Wiley, 2011.

Marques Filho, O. e Vieira Neto, H. **Processamento digital de imagens.** Brasport, Rio de Janeiro, 1999.

Matsumoto, E. Y. **MATLAB® 7 Fundamentos**. Érica, São Paulo, 2006.

Medina, B. V. O. **Sistema de visão computacional aplicado a um robo cilíndrico acionado pneumaticamente**. Dissertação de Mestrado, Universidade Federal Do Rio Grande do Sul, 2015.

Moeslund, T. B. **Introduction to Video and Image Processing**. Springer, 2012.

Nasr-Isfahani, S., Mirsafian, A. e Masoudi-Nejad, A. A New Approach for Touching Cells Segmentation. **International Conference on BioMedical Engineering and Informatics**, p. 816-820, 2008.

Otsu, N. A Threshold Selection Method from Gray Level Histograms. **IEEE Transactions on Systems, Man and Cybernetics**, Vol. 1 SMC-9, p. 62-66, 1979.

Paluszek, M. e Thomas, S. **MATLAB® Recipes: A Problem-Solution Approach**. Apress, 2015.

Qin, Y., Wang, W., Liu, W. e Yuan, N. Extended-Maxima Transform Watershed Segmentation Algorithm for Touching Corn Kernels. **Advances in Mechanical Engineering**, 2013.

Ren, S., He, K., Girshick, R. e Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, vol. 39, n° 6, p. 1137-1149, 2017.

Romano, V. F. **Robótica industrial: aplicação na indústria de manufatufatura e de processos**. Edgard Blucher, 2002.

Sansoni, G., Bellandi, P., Leoni, F. e Docchio, F. Optoranger: A 3D pattern matching method for bin picking applications. **Optics and Lasers in Engineering**, vol. 54, p. 222-231, 2014.

Scavino, E., Wahab, D. A., Basri H., Mustafa M. M. e Hussain A. An Efficient Segmentation Technique for Known Touching Objects Using a Genetic Algorithm Approach. **AI 2007: Advances in Artificial Intelligence**, vol. 4830, p. 786-790, 2007.

Shih, F. Y. **Image processing and pattern recognition: fundamentals and techniques**. Wiley, 2010.

Siciliano, B., Sciavicco, L., Villani, L. e Oriolo, G. **Robotics: Modelling, Planning and Control.** Springer, 2009.

Spong, M. W., Hutchinson, S. e Vidyasagar, M. **Robot Modeling and Control.** John Wiley & Sons, 2006.

STEEMER IMAGING. **Backlight illumination – diffuse**. https://www.stemmer-imaging.co.uk/en/knowledge-base/backlight-illumination-diffuse/. Acesso em: 18-10-2018.

Uijlings, J.R.R., van de Sande, K.E.A., Gevers, T. e Smeulders, A. W. M. Selective Search for Object Recognition. **International Journal of Computer Vision**, vol. 104,iIssue 2, p. 154–171, 2013.

Wang, Y., Li, S., Jia, M. e Liang, W. Viewpoint Estimation for Objects with Convolutional Neural Network Trained on Synthetic Images. **PCM 2016: Advances in Multimedia Information Processing. Lecture Notes in Computer Science**, vol. 9917, p. 169-179, 2016.

BIBLIOGRAFIA CONSULTADA

Axis Communications. **CCD and CMOS sensor technology.** Technical white paper. https://www.axis.com/files/whitepaper/wp_ccd_cmos_40722_en_1010_lo.pdf. Acesso em: 24-09-2018.

Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S. e Lew, M. S. Deep learning for visual understanding: A review, **Neurocomputing**, vol. 187, p. 27-48, 2016.

He, K., Gkioxari, G., Dollár, P. e Girshick, R. Mask R-CNN**. IEEE International Conference on Computer Vision (ICCV)**, p. 2980-2988, 2017.

Jähne, B. **Digital Image Processing.** Springer, 2005.

Li, G., Hou, Y. e Wu, A. Fourth Industrial Revolution: Technological Drivers, Impacts and Coping Methods. **Chinese Geographical Science**, 2017.

Liu, M., Tuzel, O., Veeraraghavan, A., Taguchi, Y., Marks, T. K. e Chellappa, R. Fast object localization and pose estimation in heavy clutter for robotic bin picking. **The International Journal of Robotics Research**, vol. 31, issue 8, p. 951–973, 2012.

Mishra, M. e Srivastava, M. A View of Artificial Neural Network. **International Conference on Advances in Engineering & Technology Research**, 2014.

Oh, J., Lee, S. e Lee, C. Stereo Vision Based Automation for a Bin-Picking Solution. **International Journal of Control, Automation, and Systems**, vol. 10, issue 2, p. 362-373, 2012.

Pérez, L., Rodríguez, I., Rodríguez, N., Usamentiaga, R. e García, D. F. Robot Guidance Using Machine Vision Techniques in Industrial Environments: A Comparative Review. **Sensors**, vol. 16, issue 3, p. 335, 2016.

Radhakrishnamurthy, H. C., Ramachandran, P. M. N. e Yaacob, S. Stereo vision system for a bin picking adept robot. **Malaysian Journal of Computer Science**, vol. 20, issue 1, 2007.

Raskar, R., Tan, K., Feris, R., Yu, J. e Turk, M. Non-photorealistic Camera: Depth Edge Detection and Stylized Rendering using Multi-Flash Imaging. **ACM Transactions on Graphics**, vol. 23, issue 3, p. 679–688, 2004.

Russell, S. J. e Norvig, P. **Inteligência artificial.** Elsevier, Rio de Janeiro, 2013.

Szeliski, R. **Computer Vision: Algorithms and Applications**. Springer, 2010.

Veen, F. V. **The Neural Network Zoo – The Asimov Institute**. http://www.asimovinstitute.org/neural-network-zoo/, 2016, Acesso em: 10-10-2018.

Weeks, A. R. **Fundamentals of Electronic Image Processing**. SPIE Optical Engineering Press, Washington, 1996.

ANEXO I – Arquiteturas utilizadas em redes neurais

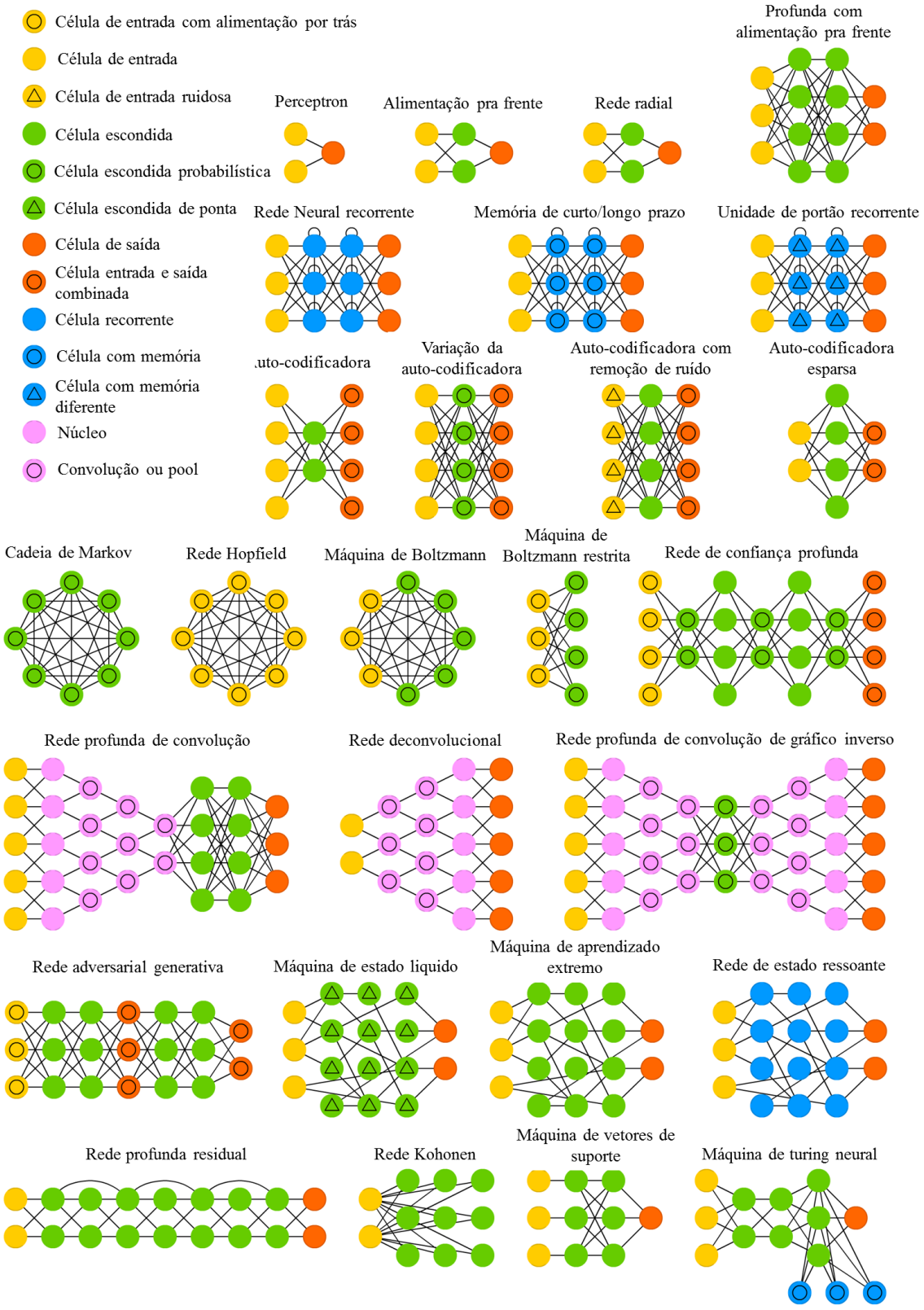


Figura I.1 – Arquitetura das redes neurais empregadas