APLICAÇÃO DE TÉCNICAS INTELIGENTES PARA CONTROLE E AUTOMAÇÃO DE DISPOSITIVO ROBÓTICO VIRTUAL[1]

Mailson Gonçalves da SILVA[2]
Graduado em Engenharia de Controle e Automação
IFSP/Câmpus São Paulo

Muriell de RODRIGUES E FREIRE[3]
Doutor em Engenharia Elétrica /UNIFEI
Docente da Área de Engenharia
IFSP/Câmpus São João da Boa Vista

RESUMO
A inteligência artificial vem sendo cada vez mais utilizada nos sistemas industriais a fim de torná-los mais inteligentes e flexíveis às alterações do ambiente. Atualmente, a indústria demanda equipamentos cada vez mais autônomos em suas linhas de produção para desempenhar tarefas que são complexas ou repetitivas de serem executadas por seres humanos, consistindo em uma linha de pesquisa que continua a ser explorada. Este trabalho tem como objetivo aplicar redes neurais artificiais e visão computacional para geração de trajetória e na orientação de um manipulador robótico em um ambiente virtual utilizando a plataforma V-REP. Para desenvolver o projeto foi utilizada a biblioteca OpenCV para a linguagem C#, onde os algoritmos de inteligência e supervisório foram desenvolvidos. No V-REP foi construído o ambiente industrial com robô e esteiras. Com o MatLab, foram levantados os dados para treinamento da rede neural a ser implementada. Os resultados obtidos mostraram que o uso de algoritmos de inteligência para orientação e geração de trajetórias em uma tarefa de *pick-and-place* obteve um bom desempenho no quesito flexibilidade, atingindo o objetivo proposto. Observou-se, porém, algumas limitações no sistema - amenizáveis com a modificação de algumas das metodologias aplicadas.
Palavras-chave: Robótica; Rede Neural Artificial; Visão Computacional; *Pick-and-place*; Simulação.

Introdução

A inteligência artificial conquistou espaço nos mais diversos setores nos últimos anos, sendo, assim, uma área com grandes perspectivas de crescimento para o futuro. Na indústria, a utilização de linhas de produção inteligentes e integradas inaugurou uma nova era, a 4ª Revolução Industrial (a chamada Indústria 4.0). Cada vez mais busca-se

[1] Artigo resultante de Trabalho de Conclusão de Curso em Engenharia de Controle e Automação. Orientador Prof. Dr. Muriell de Rodrigues e Freire.
[2] Endereço eletrônico: mailson96@gmail.com
[3] Endereço eletrônico: muriell@ifsp.edu.br

por melhor eficiência na produção, maior flexibilidade das linhas de produção, redução de refugos e maior autonomia dos maquinários. O uso de técnicas inteligentes no setor industrial tem permitido alcançar esses objetivos.

O objetivo da presente pesquisa é, a partir do uso de redes neurais e orientação por visão computacional, desenvolver um sistema de geração de trajetória para braço robótico utilizando redes neurais e orientação por visão computacional. Será utilizada a plataforma V-REP para o desenvolvimento do ambiente onde serão aplicados os algoritmos de inteligência e a linguagem C# para a criação dos algoritmos e do supervisório do sistema.

Segundo Rich e Knight (1994, p. 3), a "Inteligência Artificial (IA) é a área da ciência da computação orientada ao entendimento, construção e validação de sistemas inteligentes, isto é, sistemas que exibem, de alguma forma, características associadas ao que chamamos inteligência". Ao longo do tempo, foram desenvolvidos vários algoritmos e conceitos nessa área - dentre eles, as redes neurais artificiais e a visão computacional, que serão utilizadas neste trabalho.

As Redes Neurais Artificiais (RNA) foram desenvolvidas com base no neurônio do cérebro humano e consistem numa estrutura de vários neurônios artificiais que emitem um sinal de saída quando a combinação linear de suas entradas excede um valor de limiar (RUSSEL; NORVIG, 2013).

Essas redes podem ser estruturadas em várias arquiteturas. No caso desta pesquisa, será usada a estrutura *feedforward*, mais especificamente a rede Perceptron de Múltiplas Camadas (PMC) com treinamento supervisionado via algoritmo *backpropagation*. A Figura 1 mostra o modelo matemático de um neurônio simples.

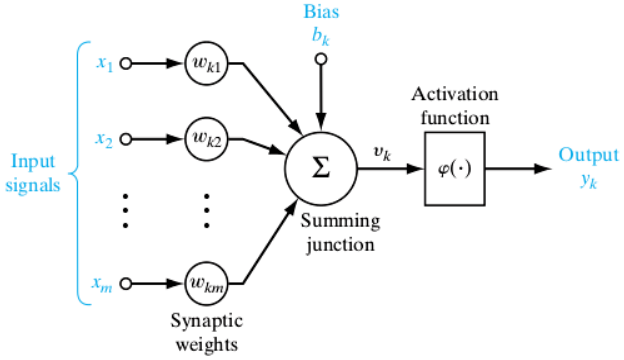

Figura 1: Modelo básico de neurônio artificial
Fonte: Mannarswamy (2017)

A rede PMC é composta por ao menos uma camada oculta entre as camadas de entrada e saída. Desta forma, possui ao menos duas camadas de neurônios e é tida como uma das mais versáteis.

Nessa configuração, os sinais da camada de entrada são propagados ao longo da rede até alcançar a camada de saída sempre em uma única direção. Diferentemente de outros tipos de rede, a rede PMC pode conter mais de um neurônio na sua camada de saída (SILVA; SPATTI; FLAUSINO, 2010).

Para ocorrer a aprendizagem na rede PMC, esta deve passar por um processo de treinamento supervisionado. Para isso se utiliza o algoritmo *backpropagation* ou regra Delta generalizada para o treinamento. De forma resumida, esse algoritmo é executado em duas partes. Na primeira, são obtidas as saídas de acordo com os sinais de entrada sem alteração dos pesos sinápticos e limiares. Na segunda, é obtido o desvio da resposta calculada com relação à desejada. Ocorre, com esse valor de erro, o ajuste dos pesos sinápticos e da função de limiar da camada de saída em direção à camada de entrada (SILVA; SPATTI; FLAUSINO, 2010).

A visão computacional é um campo da computação que utiliza técnicas de extração de informações em imagem para utilização dessas informações em tomadas de decisão e realização de ações (SHAPIRO; STOCKMAN, 2000). As etapas básicas que compõem um algoritmo de visão computacional são apresentadas na Figura 2.

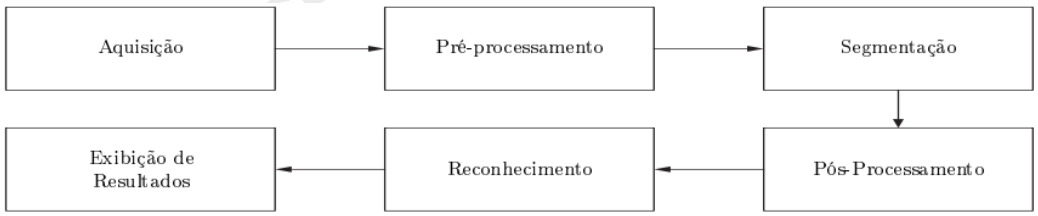

Figura 2: Etapas de um sistema de visão computacional
Fonte: Costa (2014)

Resumidamente, o processamento de imagem ocorre da seguinte forma: inicialmente é feita a aquisição da imagem pela câmera, e, em seguida, essa imagem é submetida a um pré-processamento para se retirar ruídos e ajustar contraste, a fim de melhorar a qualidade da imagem para a próxima etapa. Após isso, acontece a segmentação para identificar os objetos de interesse e, caso seja necessário, há um pós-

processamento para melhorar a qualidade da segmentação. Por fim, os objetos segmentados são classificados para o reconhecimento do objeto, e, então, os resultados do processamento são exibidos (COSTA, 2014).

Contudo nem sempre todas as etapas são realizadas. Quando o interesse é fazer apenas a detecção de um objeto, como é o caso desse trabalho, a etapa de reconhecimento não é feita.

Para implementação da visão computacional no projeto, a biblioteca OpenCV desenvolvida pela Intel foi utilizada. De acordo com o próprio website da biblioteca:

> O OpenCV (*Open Source Computer Vision Library*) é uma biblioteca de software de visão computacional e aprendizado de máquina em código aberto. O OpenCV foi construído para fornecer uma infraestrutura comum para aplicações de visão computacional e para acelerar o uso da percepção de máquina nos produtos comerciais. (…) A biblioteca é amplamente utilizada em empresas, grupos de pesquisa e órgãos governamentais (OPENCV, [201?], tradução nossa).[4]

Segundo Rosário (2010) "um manipulador robótico é um dispositivo que tem por função posicionar e orientar uma ferramenta dedicada para uma operação fixa no seu elemento terminal". Para a correta execução de movimentos do manipulador para um local desejado, é necessário determinar as cinemáticas direta e inversa de um robô.

A cinemática direta tem como objetivo determinar, de acordo com um dado conjunto de ângulos de juntas (CRAIG, 2012), a posição e a orientação do efetuador final no espaço cartesiano em relação ao sistema de coordenadas da base.

A cinemática inversa busca obter todos os possíveis ângulos de junta para atingir uma certa posição e orientação no espaço cartesiano. Porém, não se trata de um problema simples. Por ter característica não linear, a solução pode não existir ou então existirem várias soluções para uma mesma posição e orientação (CRAIG, 2012).

Contudo, para alcançar um movimento coordenado do manipulador, as equações de cinemática não bastam. Para se conseguir um deslocamento suave e controlado do robô, é necessário que os ângulos das juntas variem de acordo com uma função de

[4] OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. (…) The library is used extensively in companies, research groups and by governmental bodies.

tempo a fim de descrever uma trajetória entre o ponto de partida e o de chegada. A determinação desta função é o objetivo da geração de trajetória (CRAIG, 2012) que será realizada neste trabalho através do uso de redes neurais.

Contudo, para ensinar a rede neural como planejar uma trajetória, é necessário, primeiramente, gerar exemplos de trajetórias para treinamento da RNA. Para isso, foi utilizado o planejamento de trajetória no espaço de juntas. Nessa abordagem, são determinados, para cada junta do robô, os coeficientes de um polinômio, que determina os valores de ângulo de junta em função do tempo para executar uma trajetória entre um ponto inicial e o final desejado. É necessário fornecer o tempo de duração desejado da trajetória e os pontos inicial e final (CRAIG, 2012).

O grau do polinômio depende do número de restrições a serem atendidas, como as posições inicial e final e velocidades inicial e final do robô na trajetória. Esse método de interpolação foi escolhido pois melhor atende o objetivo do trabalho, sendo, além disso, mais simples de computar e também não gerar problemas com singularidade (CRAIG, 2012).

Para acelerar o desenvolvimento da pesquisa e reduzir custos, os algoritmos desenvolvidos não serão aplicados em um robô ou ambiente físico, mas virtual. Para isso, a plataforma de simulação V-REP – que é um ambiente integrado de desenvolvimento utilizado para simulação de automação industrial, criação rápida de algoritmos e prototipagem e para ensino de robótica, permitindo o controle da simulação no próprio software ou em aplicação externa (COPPELIA ROBOTICS, [201?]) – foi usada.

A motivação da realização deste trabalho está baseada na grande exploração do tema ainda atualmente e na constante busca por se desenvolverem dispositivos, para automação, que sejam flexíveis na realização de tarefas, diminuindo o trabalho e a necessidade de intervenção humana em uma operação. Ao final do desenvolvimento, ao realizar o processo de geração de trajetória através de algoritmos de inteligência aplicados a robótica, espera-se obter um sistema com maior autonomia na execução de tarefas.

Para atingir o objetivo proposto, o tema será desenvolvido em duas etapas. Na primeira etapa, o foco é a familiarização com as ferramentas a serem utilizadas e a criação do algoritmo de visão computacional. Na segunda etapa, ocorrerá o

desenvolvimento da rede neural para geração de trajetória. Na seção a seguir, a metodologia empregada será detalhadamente apresentada.

Desenvolvimento

A tarefa a ser executada pelo manipulador robótico consiste em um *pick-and-place*. Nela, o objeto presente numa esteira de entrada será identificado, considerando sua forma geométrica (quadrado, círculo ou triângulo), e em seguida colocado no seu devido lugar na esteira de saída.

Para o desenvolvimento deste projeto utilizou-se os softwares: Microsoft Visual Studio C# 2010 Express, MatLab 2013, Pacote Robotics Toolbox para MatLab, Simulador V-Rep e Biblioteca OpenCV para C# (EmguCV). Como o projeto foi concebido em ambiente virtual, não houve custos financeiros para sua realização.

O trabalho foi divido em duas etapas. Na primeira etapa, houve uma revisão de bibliografia sobre os conceitos envolvidos e um estudo das ferramentas a serem utilizadas. Em seguida, foi desenvolvido um ambiente virtual no V-REP para os testes iniciais e implementado um supervisório em C# para comunicação com V-REP, com algoritmo de visão computacional e comandos para controle manual de um robô.

Para realizar a detecção dos objetos no cenário com a visão computacional, inicialmente, foi necessário aplicar filtros de pré-processamento para melhorar a qualidade da segmentação e também para diminuir a influência de alguns componentes da imagem no resultado final. Após vários testes, os filtros que geraram os melhores resultados para a aplicação foram: o filtro da mediana, filtro gaussiano e o filtro bilateral que foram utilizados em conjunto na etapa de pré-processamento.

A segmentação e detecção dos objetos, foi realizada através da Transformada de Hough (HOUGH,1962) e do Detector de bordas de Canny.

No V-REP, foi criado um cenário de teste para melhor compreensão da ferramenta e para testar a comunicação com o Visual Studio. Neste cenário, foi inserido um robô, esteiras, sensor de presença, câmera e peças de diferentes formatos.

Um supervisório em C# foi programado para estabelecer comunicação com o ambiente de simulação. Para isso, foi necessário utilizar um *wrapper* (classe cujas funções estão implementadas em outro lugar) que permitisse a utilização dos códigos

necessários para a conexão da aplicação C# com o V-REP e para a manipulação dos objetos no ambiente virtual via Remote API (componente do V-REP que permite o controle da simulação via aplicação externa).

Nesse supervisório, através do valor dos ângulos das juntas, foi implementada a programação para controlar o manipulador de maneira manual e fazer a captura da imagem da câmera. O algoritmo de detecção de objetos criado anteriormente também foi inserido no programa para teste, incluindo a obtenção do centroide dos objetos detectados para sua localização no ambiente.

Na segunda etapa do projeto foram realizadas as seguintes atividades:

- Elaboração do cenário definitivo de simulação no V-REP;
- Calibração das coordenadas da câmera;
- Levantamento do espaço de trabalho do robô;
- Geração de trajetórias para conjunto de treinamento da RNA;
- Criação e treinamento do algoritmo RNA para geração de trajetória com C#;
- Programação da rotina responsável pela tarefa de *pick-and-place*.

Inicialmente, na segunda etapa, foi construído no V-REP o cenário final a ser usado no trabalho, que pode ser visto na Figura 3.

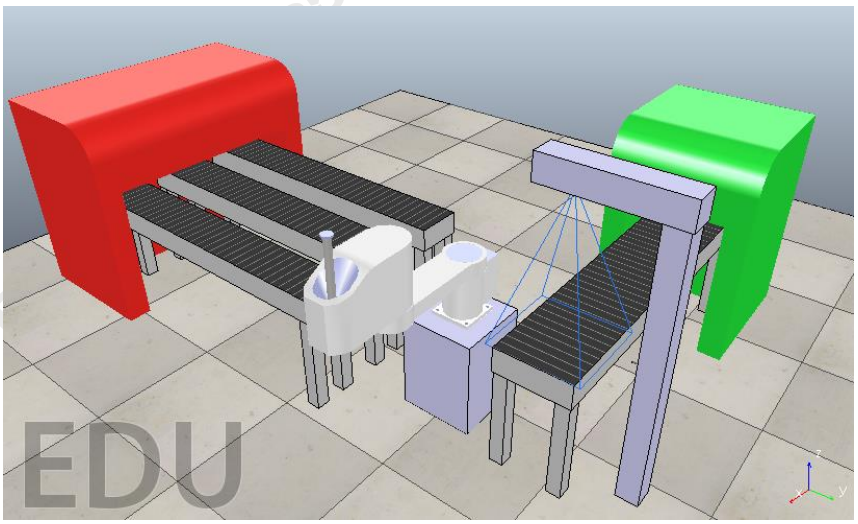

Figura 3: Cenário final construído no V-REP
Fonte: Autores

As peças a serem separadas são geradas e posicionadas aleatoriamente sobre a esteira que transporta o objeto até o robô. Isto foi feito via programação no C# e V-REP.

Em seguida, a calibração da câmera foi realizada com o propósito de converter as coordenadas do centroide do objeto detectado que estão em valores de pixel para coordenadas reais do ambiente V-REP. Como a câmera foi posicionada de modo a ter uma visão superficial (2D) do objeto, a calibração apenas das coordenadas X e Y foi feita. Para isso, foi necessário apenas conhecer os valores extremos das coordenadas reais em X e Y que a câmera é capaz de capturar (Figura 4) e a resolução da câmera. Com isso, podem-se estabelecer as relações matemáticas (1) e (2) que definem a calibração.

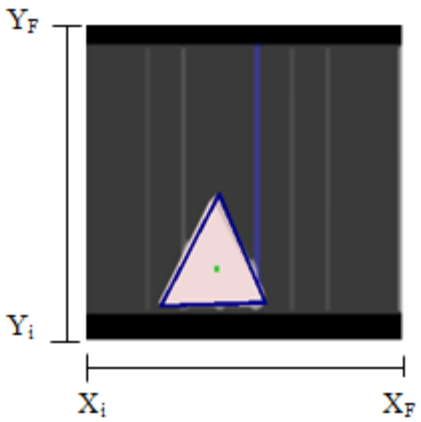

Figura 4: Calibração do sensor de visão do V-REP
Fonte: Autores

$$X = X_i + \left(\frac{X_F - X_i}{Res} * C_x\right) \quad (1)$$

$$Y = Y_F - \left(\frac{Y_F - Y_i}{Res} * C_y\right) \quad (2)$$

Em que C_x e C_y são as coordenadas do centroide em pixels, X e Y são o resultado da calibração em cada eixo e Res diz respeito à resolução da câmera, que, no caso do sensor de visão do V-REP, é de 256x256.

Para determinar o espaço de trabalho do robô SCARA, primeiramente, foi necessário definir a cinemática do manipulador. Com a notação DH, se obteve a cinemática direta. Utilizando o método analítico sobre a matriz de transformação resultante, obteve-se a cinemática inversa.

Conhecendo os limites de ângulo de cada junta, foi calculado o alcance máximo e mínimo em cada eixo do espaço através da cinemática direta. Com esses valores, foi possível determinar os pontos no espaço que o robô consegue, efetivamente, atingir através do uso da cinemática inversa. Assim, foi mapeado o espaço de trabalho do manipulador, com uso do MatLab e do pacote Robotics Toolbox.

Na atividade quatro da segunda etapa, efetuou-se a geração de trajetórias para treinamento da RNA. Para isso, foi gerado um total de 38 trajetórias para compor o conjunto de treinamento. Todas foram geradas no espaço de juntas, com uma duração fixada em cinco segundos.

As trajetórias geradas levaram em consideração apenas as juntas rotacionais, ou seja, apenas o posicionamento no plano X-Y. A última junta foi manipulada à parte, no momento da execução da tarefa de *pick-and-place*, para atingir a posição desejada no eixo Z. Todo o procedimento, por meio do MatLab e do pacote Robotics Toolbox, foi realizado. Os dados de entrada e saída para treinamento foram normalizados entre 0 e 1 e armazenados em forma de planilha do Excel para posterior utilização no Visual Studio.

O algoritmo de rede neural foi pensado para gerar a trajetória ponto a ponto, recebendo como entrada os pontos inicial e final desejados, não considerando: a coordenada do ponto em Z, e o tempo que é incrementado em passos de 0,1 segundos a cada saída gerada até atingir o tempo de duração total da trajetória. E como saída, a RNA fornece os ângulos de junta que correspondem ao próximo ponto da trajetória dentro da área de trabalho do robô. A Figura 5 ilustra essa ideia:

Figura 5: Esquema da RNA desenvolvida
Fonte: Autores

A estrutura da RNA foi desenvolvida no Visual Studio C# . O treinamento fez uso das trajetórias geradas no MatLab com a atualização dos pesos sinápticos ocorrendo padrão a padrão. Parâmetros do algoritmo, como números de neurônios e camadas, foram definidos empiricamente.

Para realizar a tarefa de *pick-and-place*, foi programada em C# uma rotina que faz uso do algoritmo de visão computacional e da rede neural desenvolvidos para comandarem o robô. O manipulador no início e no fim da rotina está posicionado em uma posição inicial chamada de *home*, que serve como ponto de partida das ações do robô. Na Figura 6, contendo as etapas a serem executadas para cada objeto presente na esteira, é apresentado o fluxograma da rotina.

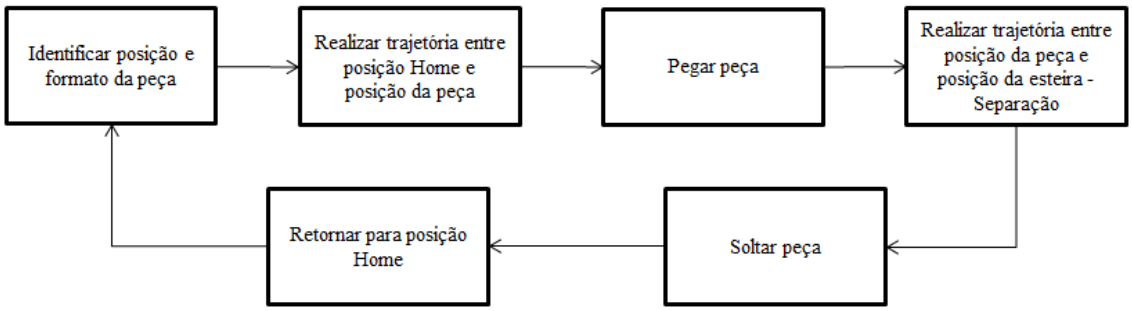

Figura 6: Fluxograma da rotina para tarefa de *pick-and-place*
Fonte: Autores

Depois de realizada todas as atividades da segunda etapa, foram feitos testes e medições com o sistema obtido. Os resultados serão apresentados e discutidos na próxima seção.

Resultados obtidos

Para realizar a comunicação entre Visual Studio e V-REP, foi desenvolvido um supervisório em C#. A Figura 7 mostra a aplicação desenvolvida:

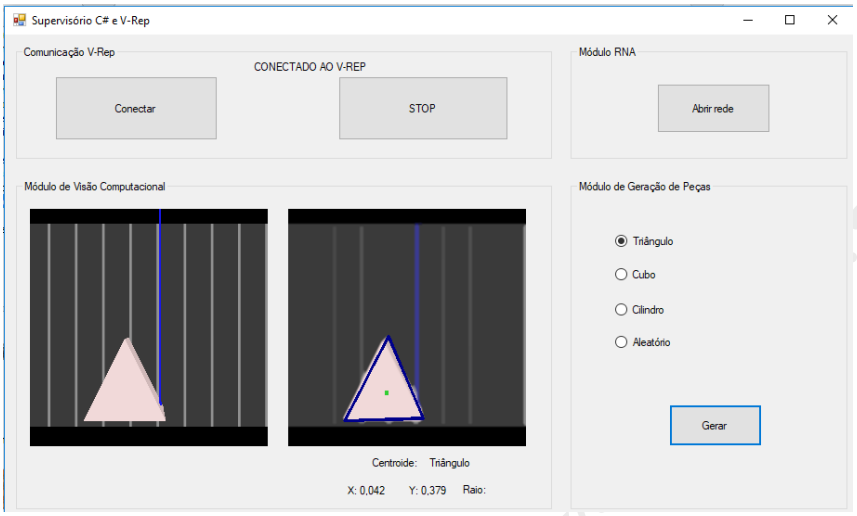

Figura 7: Supervisório desenvolvido para comunicação com V-REP
Fonte: Autores

Por meio deste supervisório, foi possível realizar a leitura dos sensores do ambiente virtual, atuar sobre esteiras e robô e também implementar os algoritmos de visão computacional e de rede neural para execução da tarefa proposta.

No algoritmo de visão computacional, por meio de testes, foram obtidos os parâmetros utilizados para segmentação. Na Figura 8, é apresentado o resultado de detecção de um dos testes realizados:

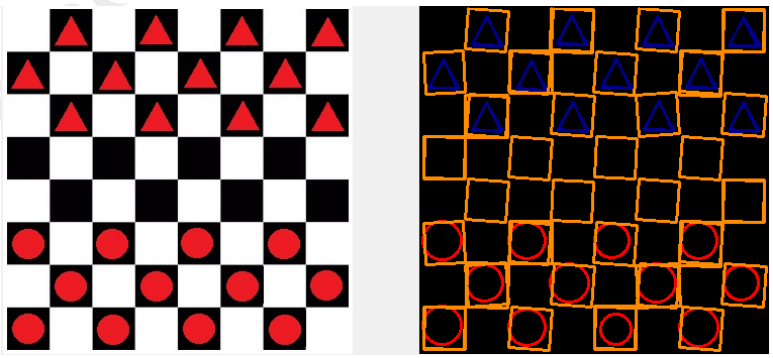

Figura 8: Resultado da detecção de quadrados, triângulos e círculos em um dos testes
Fonte: Autores

A detecção dos círculos foi obtida com a transformada de Hough, enquanto os retângulos e triângulos, pelo filtro de Canny.

Para que fosse possível localizar o objeto detectado no ambiente, foi necessário obter o centroide da forma detectada. No caso dos círculos e quadrados, este valor foi obtido facilmente, pois a programação responsável pela detecção já fornece essa informação. Contudo, para os triângulos, para obter a coordenada X e Y do centroide, foi necessário realizar o cálculo das coordenadas médias utilizando os vértices do triângulo.

A calibração da câmera foi feita seguindo a metodologia descrita anteriormente. Contudo, foi observado, nos testes, um nível de erro grande para algumas posições da peça, sendo necessário realizar um ajuste nas equações de calibração de acordo com os resultados experimentais para contornar o problema. Isso ocorreu devido à influência dos filtros de pré-processamento, que geram uma distorção na imagem ao retirar ruídos e prepará-la para segmentação.

Concluído os ajustes na calibração, foram realizadas várias medidas para estimar o desvio percentual médio da conversão. Essas medidas, juntamente com o desvio médio obtido, estão na Tabela 1.

Peça	Posição real		Câmera		Desvio (%)	
	X	Y	X	Y	X	Y
Prisma Triangular	0,0492	0,4863	0,047	0,481	4,472	1,090
Cubo	0,0495	0,393	0,047	0,388	5,051	1,272
Cilindro	0,0489	0,3674	0,044	0,361	10,020	1,742
Prisma Triangular	0,0492	0,3948	0,049	0,368	0,407	6,788
Cubo	0,046	0,3636	0,046	0,353	0,000	2,915
Cilindro	0,0495	0,4204	0,045	0,421	9,091	0,143
Prisma Triangular	0,0492	0,5013	0,047	0,496	4,472	1,057
Cubo	0,046	0,4281	0,043	0,431	6,522	0,677
Cilindro	0,0489	0,3534	0,045	0,344	7,975	2,660
Prisma Triangular	0,0457	0,3759	0,047	0,346	2,845	7,954
Cubo	0,0495	0,3728	0,045	0,364	9,091	2,361
Cilindro	0,0454	0,4975	0,046	0,522	1,322	4,925
Cubo	0,0495	0,4063	0,048	0,405	3,030	0,320
Cilindro	0,0454	0,3711	0,044	0,364	3,084	1,913
Cubo	0,046	0,5046	0,045	0,521	2,174	3,250
Prisma Triangular	0,0492	0,4745	0,048	0,463	2,439	2,424
Prisma Triangular	0,0492	0,3608	0,049	0,331	0,407	8,259
				Média	**4,259**	**2,927**

Tabela 1: Desvio entre a posição real do objeto e a obtida pela calibração
Fonte: Autores

Assim, obteve-se um desvio médio de cerca de 4% em X e de 3% em Y, entre as coordenadas da posição real do objeto e a que foi obtida pela calibração. Sendo, dessa forma, um valor de desvio aceitável. Além disso, todas as peças utilizadas foram identificadas, atingindo, por esta via, 100% de eficiência na detecção dos objetos.

Utilizando o MatLab, foi obtido, representado na Figura 9, o modelo geométrico do robô SCARA, e também as equações da cinemática direta que apresentaram o formato exposto nas fórmulas (3) a (5).

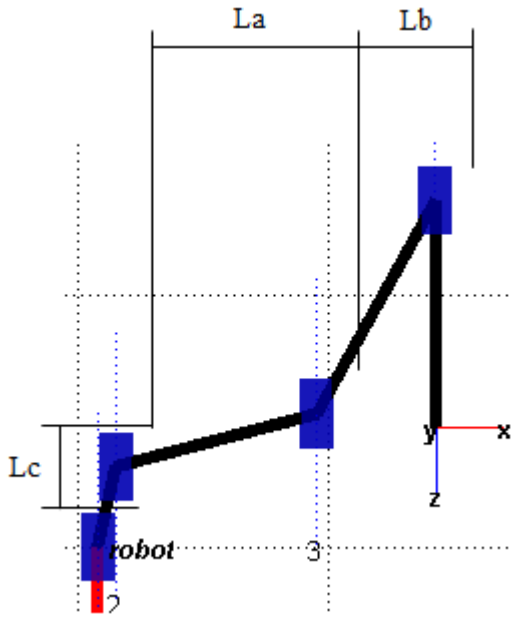

Figura 9: Modelo geométrico do robô SCARA
Fonte: Autores

$$P_X = L_b * cos\ cos\ (\theta_1 + \theta_2 + c_1) + L_a * cos\ cos\ (\theta_1 + c_1) + c_2 \quad (3)$$

$$P_Y = L_b * sen(\theta_1 + \theta_2 + c_1) + L_a * (\theta_1 + c_1) - c_3 \quad (4)$$

$$P_Z = L_C - d_3 \quad (5)$$

Em que P_X, P_Y e P_Z representam a posição do atuador final; θ_1, θ_2 e d_3 dizem respeito às juntas rotacional e prismática, respectivamente, que formam o robô, e c_1, c_2, c_3 são constantes que se originaram do processo de obtenção da cinemática direta.

Depois de obtida a cinemática direta do robô, obteve-se, também, a cinemática inversa por meio de manipulações algébricas das equações de (3) a (5). Utilizando a

cinemática inversa e os limites dos ângulos das juntas, a área de trabalho do robô (Figura 10) foi levantada.

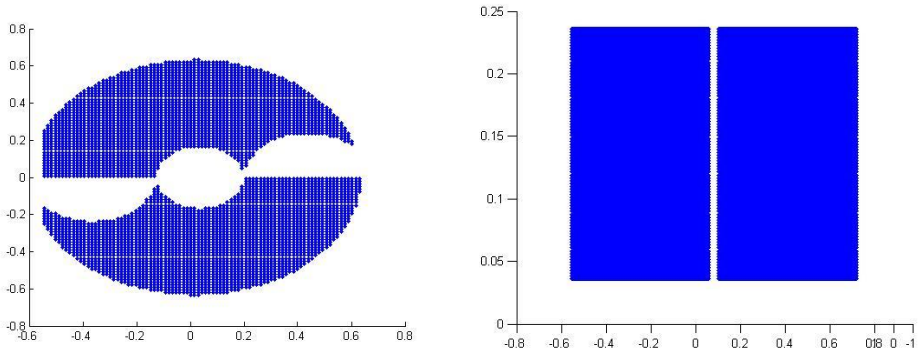

Figura 10: Espaço de trabalho do robô SCARA
Fonte: Autores

Para realizar o treinamento da rede neural, foi utilizado um notebook com sistema operacional Windows 10 Pro de 64 bits, com 4 GB de memória RAM e processador Intel Core i5-3340M de 2,70Ghz.

A definição da arquitetura da RNA foi obtida de forma empírica. Após a realização de vários testes, obteve-se a configuração de neurônios 60-50-50-02, que apresentou melhor desempenho. Os parâmetros definidos para o treinamento resultaram em uma taxa de aprendizagem de 0,08 e um termo de momentum de 0,9 com um total de 400.000 interações ou épocas. O processo de treinamento levou cerca de 2 dias para ser completado.

Após o treinamento, a RNA foi aplicada na tarefa de *pick-and-place* proposta.

Durante a realização da tarefa, o algoritmo teve de gerar trajetórias utilizando posições não conhecidas que foram fornecidas pela visão computacional, efetuando, também, a separação das peças. Neste cenário, foram feitos vários testes para calcular o desvio percentual médio de posicionamento do robô com a rede neural em relação à posição da peça a ser manipulada. Com isso, também foi obtida uma medida da capacidade de generalização da RNA desenvolvida. Os resultados obtidos estão registrados na Tabela 2.

Peça	Posição real		Rede Neural		Desvio (%)	
	X	Y	X	Y	X	Y
Prisma Triangular	0,0492	0,4863	0,0532	0,4687	8,1301	3,6192
Cubo	0,0495	0,393	0,0544	0,3812	9,8990	3,0025
Cilindro	0,0489	0,3674	0,0541	0,3514	10,6339	4,3549
Prisma Triangular	0,0492	0,3948	0,055	0,3622	11,7886	8,2573
Cubo	0,046	0,3636	0,0546	0,3446	18,6957	5,2255
Cilindro	0,0495	0,4204	0,0533	0,4127	7,6768	1,8316
Prisma Triangular	0,0492	0,5013	0,0533	0,4813	8,3333	3,9896
Cubo	0,046	0,4281	0,0524	0,4209	13,9130	1,6819
Cilindro	0,0489	0,3534	0,0545	0,3349	11,4519	5,2349
Prisma Triangular	0,0457	0,3759	0,0548	0,3382	19,9125	10,0293
Cubo	0,0495	0,3728	0,0542	0,3552	9,4949	4,7210
Cilindro	0,0454	0,4975	0,0535	0,5018	17,8414	0,8643
Cubo	0,0495	0,4063	0,0544	0,3991	9,8990	1,7721
Cilindro	0,0454	0,3711	0,054	0,3544	18,9427	4,5001
Cubo	0,046	0,5046	0,053	0,5004	15,2174	0,8323
Prisma Triangular	0,0492	0,4745	0,0537	0,4537	9,1463	4,3836
Prisma Triangular	0,0492	0,3608	0,0555	0,3247	12,8049	10,0055
				Média	**12,5754**	**4,3709**

Tabela 2: Desvio entre a posição real do objeto e o posicionamento do robô com RNA
Fonte: Autores

O desvio médio obtido foi de cerca de 12,6% na coordenada X e de 4% na coordenada Y. Esse nível de erro é devido ao desvio herdado da visão computacional unido ao desvio da própria rede neural, uma vez que, para valores de entrada desconhecidos, a rede neural procura extrapolar o seu aprendizado, obtido no treinamento, para calcular uma saída aproximada para um valor de entrada novo. Apesar de um erro maior no eixo X, o robô foi capaz de manipular todos os objetos envolvidos nos testes, efetuando sua separação de acordo com a forma geométrica, resultando assim em uma eficiência de 100% e uma capacidade de generalização satisfatória.

Os resultados obtidos e observados nos testes revelaram algumas vantagens e desvantagens do sistema desenvolvido.

A primeira vantagem é com relação à geração de trajetória utilizando uma rede neural. O método do espaço de juntas, empregado para gerar o conjunto de treinamento, possui o problema de utilizar o cálculo da matriz inversa, que possui um custo

computacional alto para ser resolvido, sendo um problema caso este método fosse utilizado para gerar trajetória em tempo real em uma aplicação.

Outra vantagem é o aumento da flexibilidade do robô ao utilizar visão computacional em conjunto com uma rede neural, permitindo que o sistema consiga se adaptar a mudanças na posição do objeto de interesse sem a necessidade de intervenção humana ou alteração da programação do sistema. Este aspecto positivo não existe com uma programação convencional de robôs.

Por fim, os resultados obtidos nos testes mostram que um sistema deste tipo é capaz de ser utilizado em uma tarefa simples de *pick-and-place*, de modo a executar um trabalho repetitivo antes realizado por um ser humano, mantendo assim o nível de produtividade e qualidade por meio da automação e, ao mesmo tempo, liberando o trabalhador para realização de tarefas mais complexas, motivadoras e que estimulem seu desenvolvimento.

Contudo, há também desvantagens. A primeira diz respeito à capacidade de generalização do algoritmo RNA desenvolvido: após vários testes foi observado que a rede neural é capaz de generalizar a geração de trajetórias apenas em regiões próximas aos dos pontos fornecidos no treinamento. Isto significa que, caso houvesse uma mudança na posição dos equipamentos, como esteira e robô, a rede neural não seria capaz de gerar trajetórias nesse novo ambiente, sendo necessário um novo treinamento. Para superar esse problema, seria necessário utilizar um conjunto de treinamento muito grande, em um nível que seria inviável para a RNA desenvolvida, ou então, talvez, utilizar outra estrutura de rede neural ao invés da PMC.

A segunda desvantagem é que o sistema desenvolvido pode não ser recomendado caso a tarefa exija uma precisão elevada, visto a porcentagem de desvio obtida. Para amenizar este problema, conjunto de treinamento poderia ser aumentado e uma câmera com maior resolução poderia ser utilizada.

Por último, o longo tempo de treinamento da rede neural. O algoritmo *backpropagation*, apesar de possuir uma boa generalização e convergência, é um algoritmo lento. Assim, outras formas de treinamento supervisionado poderiam ser testadas a fim de reduzir o tempo de treinamento permitindo, dessa forma, o uso de conjuntos de treinamento maiores.

Conclusão

O projeto buscou desenvolver um sistema de orientação e geração de trajetória a partir do uso de algoritmos de inteligência artificial a fim de proporcionar maior flexibilidade e robustez na operação de um manipulador. Utilizando ferramentas de simulação e os algoritmos de visão computacional e de rede neural PMC, esta proposta pôde ser desenvolvida e ter sua eficiência verificada em testes.

De acordo com os resultados obtidos, o objetivo inicialmente proposto pôde ser atingido. O resultado foi um sistema flexível a mudanças de posição do objeto de interesse com uma boa eficiência ao ser aplicado em uma tarefa simples de *pick-and-place*.

Entretanto houve algumas limitações que acabaram por resultar em uma robustez menor que a esperada. Isto se deu em razão da capacidade de generalização em regiões ao redor dos dados de treinamento, fazendo com que o sistema não fosse capaz de gerar uma trajetória entre pares de pontos muito distantes dos utilizados para treinamento.

Para trabalhos futuros poderiam ser realizados testes de planejamento de trajetória envolvendo formas específicas de trajetória geradas através de jacobiano, pois este método envolve cálculos de matriz pseudoinversa, que podem gerar situações de singularidade. Assim, seria interessante verificar se o uso de redes neurais nestes casos seria capaz de evitar este problema.

Além disso, como continuação desta pesquisa, o sistema desenvolvido poderia ser aplicado praticamente envolvendo um cenário real para verificar as discrepâncias entre a simulação e a realidade, fazendo também a modelagem dinâmica do robô para se obterem resultados mais fiéis aos que ocorrem na prática.

Referências

CORKE, Peter. **Robotics Toolbox**. Versão 10.2. 2017. Disponível em: http://petercorke.com/wordpress/toolboxes/robotics-toolbox. Acesso em: 08 mai. 2018.

COSTA, Rodrigo Carvalho Souza. **OpenCV.** 2014. Disponível em: http://rodcosta.eadti.com.br/tutoriais/opencv. Acesso em: 19 mai. 2018.

CRAIG, John J. **Robótica**. 3.ed. São Paulo: Pearson Education do Brasil, 2012.

EMGUCV. **EmguCV**. Versão 2.3.0.1416. 2018. Disponível em: http://www.emgu.com/wiki/index.php/Download_And_Installation. Acesso em: 08 mai. 2018.

HOUGH, Paul V.C. **Method and means for recognizing complex patterns**. Depositante: Paul V. C. Hough. US 3069654. Depósito: 25 mar. 1960. Concessão: 18 dez. 1962. Disponível em: https://patents.google.com/patent/US3069654. Acesso em: 28 abr. 2018.

MANNARSWAMY, Sandya. Everything you need to know about neural networks. **Open Source For u.com**, 16 mar. 2017. Disponível em: https://opensourceforu.com/2017/03/neural-networks-in-detail/. Acesso em: 19 mai. 2018.

MATHWORKS. **Matlab**. 2013. Disponível em: https://www.mathworks.com/products/matlab.html (versão de teste). Acesso em: 08 mai. 2018.

OPENCV. **About**. [201?]. Disponível em: https://opencv.org/about.html.Acesso em: 03 mai. 2018.

RICH, Elaine; KNIGHT, Kevin. **Inteligência Artificial**. 4. ed. São Paulo: Makron Books, 1994.

ROBOTICS, Coppelia. **Home** e **Virtual robot experimentacion platform.** Versão 3.5.0. 2018. Disponível em: http://www.coppeliarobotics.com/downloads.html. Acesso em: 08 mai. 2018.

ROSÁRIO, João Mauricio. **Robótica Industrial I:** Modelagem, Utilização e Programação. São Paulo: Baraúna, 2010.

RUSSELL, Stuart; NORVIG, Peter. **Inteligência Artificial.** 3.ed. Rio de Janeiro: Elsevier, 2013.

SHAPIRO, Linda; STOCKMAN, George. **Computer Vision**. East Lansing, MI: Pearson Prentice Hall, 2000.

SILVA, Ivan Nunes da; SPATTI, Danilo Hernane; FLAUSINO, Rogério Andrade. **Redes Neurais Artificiais para engenharia e ciências aplicadas**: curso prático. 2.ed. São Paulo: Artliber, 2010.

APPLICATION OF INTELLIGENT TECHNIQUES FOR CONTROL AND AUTOMATION OF VIRTUAL ROBOTIC DEVICE

ABSTRACT
The artificial intelligence has been increasingly used in the industrial systems in order to make them more intelligent and flexible to the environment changes. Industry currently demands increasingly autonomous equipment in its production lines to perform tasks that are complex or repetitive to be performed by humans, which is an ongoing explored line of research. This work aims to apply artificial neural networks and computer vision for trajectory generation and guidance of a robotic manipulator in a virtual environment using the V-REP platform. In order to develop the project, the OpenCV library has been used for the C# language where the algorithms of intelligence and supervisory have been developed. In the V-REP has been built the industrial environment with a robot and mats. With the MatLab the data has been lifted for the training of the neural network to be implemented. The results obtained showed that the use of intelligence algorithms for orientation and trajectory generation in a pick-and-place task achieved a good performance in the matter of flexibility, reaching the goal proposed. However, there were also some limitations in the system that can be mitigated by the modification of the methodologies applied.
Keywords*: Robotics; Artificial neural network; Computer vision; Pick-and-place; Simulation.*

Envio: novembro/2018
Aceito para publicação: março/2019