DOI: 10.48011/asba.v2i1.1097

Sistema de preensão robótica utilizando Redes Neurais Convolucionais e Primitivas Geométricas

Daniel M. de Oliveira * **Cézar B. Lemos** *
André G. S. Conceição *

* *LaR - Laboratório de robótica - Departamento de Engenharia Elétrica e da Computação, Universidade Federal da Bahia, BA, (e-mails: danielmoura@ufba.br, cezarcbl@protonmail.com e andre.gustavo@ufba.br).*

Abstract: This work presents a grasping method based on RGB+D visual information, utilizing a convolutional neural network and point cloud. The neural network, using the SSD architecture, will detect the object in RGB image and its bounding box will be used as reference to isolate the object to be grasped in the point cloud from environment objects. After detecting the object, a grasping algorithm based on object primitive geometry will be used to estimate the object pose, allowing the grasping task. To validate the system, tests in a real world environment grasping objects from a 3D printer using a UR5 robotic arm, where it was proven that the proposed algorithm allow to detect objects in the point cloud, validating the performance of the proposed system.

Resumo: Neste trabalho será apresentado um método de preensão robótica baseado em informação visual RGB+D, utilizando uma rede neural convolucional e nuvem de pontos. A rede neural, com arquitetura SSD512, detectará o objeto em imagem RGB e a área detectada na imagem será utilizada como referência na nuvem de pontos para isolar o objeto desejado do ambiente. Após o isolamento, um algoritmo de preensão robótica baseado em primitivas geométricas será aplicado para estimar a pose do objeto, permitindo a tarefa de preensão. Para validar o sistema, testes com peças feitas em uma impressora 3D e o braço robótico UR5 foram utilizados, onde foi demonstrado que a combinação dos algoritmos permite detectar objetos na nuvem de pontos, validando o desempenho do sistema proposto.

Keywords: Grasping; Point Cloud; SSD512; Deep Learning; Computer Vision; Robotic Arm.
Palavras-chaves: Preensão; Nuvem de pontos; SSD512; Aprendizado profundo; Visão Computacional; Braço Robótico.

1. INTRODUÇÃO

Com a evolução das tecnologias e o surgimento da Indústria 4.0, tarefas autônomas na indústria tem se tornado mais simples de se implementar e, como consequência, cada vez mais necessárias. Tarefas como pegar objetos em um local, navegação e produção de peças podem ser facilmente realizadas por robôs de forma autônoma, sem a necessidade de um operador. Redes de aprendizado profundo tem se popularizado em diversas áreas da indústria graças a evolução dos hardwares e algoritmos de Inteligência Artificial, principalmente na área de detecção de objetos, fazendo surgir diversos tipos de rede como YOLO (Redmon et al. (2016)), ResNet-50 (He et al. (2016)), R-CNN (Girshick (2015)) e SSD (Liu et al. (2016)). Em conjunto com essas redes, pegar objetos de forma autônoma tem se tornado cada vez mais fácil graças a avanços no hardware e nos sensores de profundidade, pois analisar a geometria das peças e localizá-las no espaço se tornou uma tarefa mais simples e barata, sem a necessidade de lasers ou diversas câmeras, permitindo ter uma noção mais realista da peça, sua localização e onde pegá-la sem necessitar de equipamentos de alto custo. Trabalhos de preensão em nuvem de pontos, como Jain and Argall (2016) e Zapata-Impata (2017), conseguem analisar a geometria da peça e estimar melhor local para pegar a peça sem conhecimento prévio da peça ou de sua forma.

Com a ideia da Indústria 4.0 e automação em mente, este artigo tem como foco desenvolver um sistema de preensão, onde será utilizado uma rede neural convolucional, utilizando a arquitetura SSD512, em uma imagem RGB e um algoritmo de detecção de primitiva geométrica na nuvem de pontos para analisar o melhor local para se pegar a peça. A implementação utilizada da rede SSD512 foi a vista em Lemos et al. (2019), pois a mesma foi feita para detectar peças desenvolvidas para ambientes industriais e se mostrou efetiva mesmo com um número pequeno de dados no treinamento. O esquema do funcionamento do sistema seguirá as seguintes etapas, conforme a Figura 1:

- Detecção do objeto em imagem RGB pela rede.
- Conversão dos pontos referentes a *bounding box* do objeto desejado em pontos da nuvem de pontos.

- Utilizar um filtro passa faixas para limitar o campo de visão na nuvem de pontos para isolar a peça de objetos do ambiente.
- Estimar a pose do objeto (posição e orientação) para efetuar a preensão.

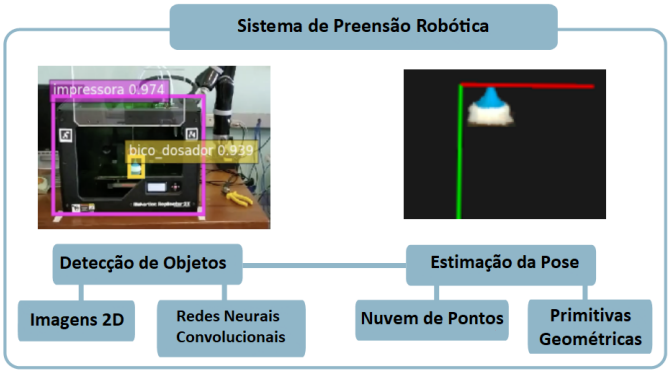

Figura 1. Sistema de preensão robótica proposto.

Para validar o sistema proposto, serão realizados diversos testes de detecção de objetos feitos em uma impressora 3D, onde estas peças são utilizadas na indústria. Após os testes de detecção, uma tarefa de pegar objetos será realizada, onde um braço robótico, no caso deste trabalho o UR5 da *universal robots*, irá pegar a peça dentro da impressora 3D, retirar da impressora e colocar em um local desejado. Como o braço não possui uma garra embutido, a garra utilizada foi o *Robotiq 2F-140* da Robotiq. Como sensor visual, a câmera de profundidade da *Intel RealSense D435* foi utilizada por possuir imagem RGB, nuvem de pontos, suporte a diversas resoluções (até 1280x720 em imagem de profundidade e até 1920x1080 em imagens RGB) e alcance mínimo da nuvem de pontos de 0.105m, ideal para tarefas de preensão onde a câmera está montada no braço robótico. A implementação do sistema utilizou o ROS, onde a rede SSD foi implementada em Python utilizando *GluonCV Toolkit* (Guo et al. (2020)), enquanto a estimação de pose e a preensão foram feitos em C++ utilizando o PCL (Rusu and Cousins (2011)).

Este artigo está organizado da seguinte maneira: A Seção 2 falará da rede de aprendizagem profunda utilizada para detectar objetos. A Seção 3 descreve o processo de preensão, sendo este separado em dois subcapítulos: detecção da peça na nuvem de pontos e o algoritmo utilizado para estimar a melhor pose para realizar a preensão. Na Seção 4 serão mostrados os resultados dos experimentos realizados para detectar objetos e pegar peças de dentro de uma impressora 3D. E por fim, na Seção 5, serão discutidos os resultados encontrados e possíveis melhorias em trabalhos futuros.

2. SSD: SINGLE SHOT MULTIBOX DETECTOR

Single Shot MultiBox Detector (SSD) é uma rede de aprendizado profundo utilizado para detecção de objetos em imagens RGB, onde, de acordo com Liu et al. (2016), possui melhor detecção em tempo real do que outras redes similares, como YOLO. Segundo testes de *benchmark* realizados por Guo et al. (2020), em questão de confiabilidade da detecção SSD perde somente para o R-CNN enquanto ganha de todos os algoritmos em desempenho. O algoritmo foi implementado neste artigo utilizando *GluonCV Toolkit* (Guo et al. (2020)) e foi modificado como visto em Lemos et al. (2019), pois o mesmo foi modificado para detectar objetos feitos em uma impressora 3D e se provou efetivo mesmo quando treinado com uma base de dados limitada.

2.1 Modelo

A SSD é uma rede neural convolucional *feed forward*, a qual produz: *bounding boxes* de tamanho fixo, um id para a classe e pontuação de confiança para cada objeto detectado, onde a rede processa a imagem somente uma vez. As camadas iniciais seguem a arquitetura chamada de padrão para detecção de objetos em imagem, segundo Liu et al. (2016), onde, para este artigo, a arquitetura padrão utilizada foi a da rede *ResNet-50*, desenvolvida por He et al. (2016). A escolha da *ResNet-50* se deu pelo seu grande desempenho em classificação, em conjunto com sua grande eficiência computacional, ficando em primeiro em diversos testes de *benchmark* realizados por Coleman et al. (2017). Na Figura 2 vemos a arquitetura final da rede quando ela utiliza VGG16 como arquitetura padrão da rede. São adicionadas estruturas auxiliares a arquitetura padrão para detectar objetos com as seguintes características:

- Mapa de características multiescala para detecção: Camadas de convolução adicionais são utilizadas para permitir detectar objetos em diferentes escalas.
- Predição convolucional para detecção: Cada camada *feature* adicionada pode produzir um número fixo de predições.
- Caixas padrão e proporções: São associadas um conjunto de *bounding boxes* com cada *feature map cell*.

2.2 Treinamento e Desempenho

A principal diferença entre treinar o SSD e outras redes de detecção está no fato de que a informação precisa ser designada para um conjunto específico de saídas de detecção. Após essa designação ser feita, a função de perda e *back propagation* são aplicadas de ponta a ponta.

Para este trabalho, a rede foi treinada com fotos de diversas peças feitas em uma impressora 3D como as vistas na Figuras 3 e 4, onde base de dados é composta por seis classes e 365 imagens RGB. Esta base de dados criada foi dívida 70% (255 imagens) para treino e o resto (110 imagens) para testes, sendo que o treinamento foi feito utilizado 60 *epochs* com indicie de aprendizado de 10^{-3} nos primeiros 20 *epochs*, 10^{-4} nos próximos 20 *epochs* e 10^{-5} no resto. Por meio de experimentos reais, foi encontrado uma exatidão de 91% na detecção, 97% de precisão e um mAP (mean average precision) de 97.5%.

3. SISTEMA DE PREENSÃO

O sistema de preensão possui em duas etapas: a primeira, eliminar objetos indesejados da nuvem de pontos utilizando a *bounding box* da rede neural, visto na Seção 3.1 e a segunda estimar uma primitiva geométrica do objeto e, baseada nesta primitiva, estimar a melhor pose de pegar o objeto, visto na Seção 3.2.

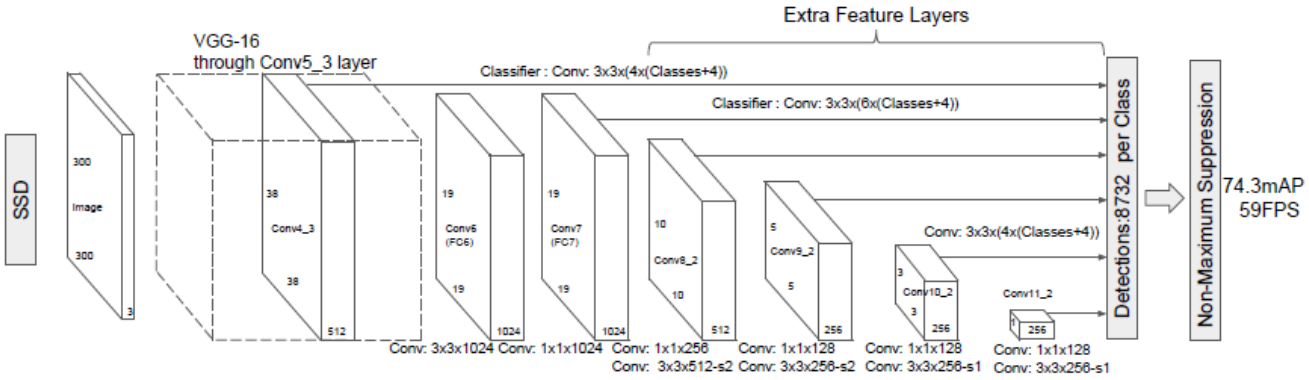

Figura 2. Arquitetura da rede SSD. Fonte: Liu et al. (2016)

Figura 3. Exemplo de peça feita na impressora 3D. Fonte: Lemos et al. (2019)

Figura 4. Peças utilizadas no treinamento e teste da rede. Fonte: Lemos et al. (2019)

3.1 Detecção de objetos na nuvem de pontos

Considere uma imagem RGB c e uma nuvem de pontos organizada p, ambas com a mesma resolução. Dentro desta imagem c, considere uma *bounding box* gerada pela rede SSD com dois pontos $c_1(u_1, v_1)$ e $c_2(u_2, v_2)$, sendo c_1 o ponto superior direito e c_2 o ponto inferior esquerdo da caixa. Assim, podemos definir os pontos $(x, y, z) \in p$ na nuvem de pontos, equivalentes aos pontos c_1 e c_2 da imagem RGB, como:

$$(x_1, y_1, z_1) = p(u_1, v_1) \tag{1}$$

$$(x_2, y_2, z_2) = p(u_2, v_2) \tag{2}$$

Com estes pontos, podemos definir os limites inferiores e superiores da *bounding box* como:

$$x_{min} = min(x_1, x_2) \tag{3}$$

$$x_{max} = max(x_1, x_2) \tag{4}$$

$$y_{min} = min(y_1, y_2) \tag{5}$$

$$y_{max} = max(y_1, y_2) \tag{6}$$

$$z_{min} = min(z_1, z_2) \tag{7}$$

$$z_{max} = max(z_1, z_2) \tag{8}$$

Com os estes limites, aplicamos um filtro passa faixas p_f, sendo a implementação feita no PCL (Rusu and Cousins (2011)), em cada um dos eixos da nuvem de pontos, limitando o campo de visão da nuvem para pontos dentro do limite inferior (x_{min},y_{min},z_{min}) e superior (x_{max},y_{max},z_{max}), assim, isolando a peça do ambiente. Após o isolamento da peça, um filtro de *Statistical Outlier Removal* foi utilizado para remover ruídos.

3.2 Estimação de primitivas geométricas

Após isolar a peça, o algoritmo de estimação de primitivas geométricas utilizado foi o demonstrado em Jain and

Argall (2016). Nele, os objetos são reduzidos a três tipos de primitivas geométricas, sendo elas: caixa, cilindro e esfera.

Para encontrar a primitiva da peça detectada, primeiro considere uma nuvem de pontos contendo somente a peça detectada como p_o com número total de pontos igual a N e um ponto $p_* \in p_o$, assim podemos definir seu centroide, p_c, como:

$$p_c = \frac{\sum_{i=1}^{N} p_i}{N}, p_i \in p_o \tag{9}$$

Com isso, podemos definir a matriz de covariância como:

$$C = \frac{1}{N} \sum_{i=1}^{N} \varepsilon_i \cdot (p_i - p_c) \cdot (p_i - p_c)^T \tag{10}$$

Onde ε_i para o ponto p_i é calculado como visto em Radu Bogdan Rusu et al. (2007). Os autovalores $\lambda_j \in R^{3x1}$ e autovetores $\boldsymbol{v}_j \in R^{3x3}$ são definidos pela seguinte relação:

$$C \cdot \boldsymbol{v}_j = \lambda_j \cdot \boldsymbol{v}_j, j \in \{1, 2, 3\} \tag{11}$$

O autovalor λ_j quantifica a variação ao longo de p_* em conjunto com o autovetor $\boldsymbol{v}_j$. Usando *PCA(Principal Component Analysis)*, podemos encontrar a variação da superfície σ_{p_*} em volta de cada ponto p_* como:

$$\sigma_{p_*} = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2} \tag{12}$$

Assim, podemos definir uma métrica Δ utilizada para determinar a melhor primitiva geométrica 3D como:

$$\Delta = \frac{M^*}{M} \tag{13}$$

Onde M^* é o número de pontos $p_* \in p_o$ para qual $\sigma_{p_*} \leq 0.01$ e M é o número de total de pontos $p_* \in p_o$.

Com o valor de Δ, podemos definir as primitivas geométricas como visto na Tabela 1.

Esférico	$0.0 \leq \Delta \leq 0.10$
Cilíndrico	$0.10 < \Delta \leq 0.40$
Caixa	$0.40 < \Delta \leq 1.0$

Tabela 1. Classificação da primitiva geométrica

Com a primitiva do objeto definida, podemos definir a posição de preensão para objetos do tipo Caixa, sendo g_p a pose final da garra e p_c o centroide do objeto, como:

$$g_p = p_c \tag{14}$$

Para objetos Esféricos, considere s_c o centro da esfera, s_r o raio da esfera e $\hat{y}$ o eixo y, assim, podemos definir a pose final para pegar a peça como:

$$g_p = s_c + s_r \cdot \hat{y} \tag{15}$$

Para objetos Cilíndricos, considere c_c o centro do cilindro, c_r o raio do cilindro e $\hat{y}$ o eixo y, assim, a pose final da garra será:

$$g_p = c_c + c_r \cdot \hat{y} \tag{16}$$

4. RESULTADOS

Os resultados são divididos em dois tipos de experimentos: Detecção de objetos, na Seção 4.1, onde serão testados os algoritmos utilizados para detectar objetos e estimar a primitiva geométrica e preensão, visto na Seção 4.2, onde uma tarefa de pegar um objeto será realizada utilizando o braço robótico *UR5*, a garra *Robotiq 2F-140* e o sensor visual *Intel RealSense D435*.

4.1 Detecção de objetos

No quesito detecção de objetos, a rede SSD consegue detectar bem os objetos com alguns falsos positivos, como visto na Figura 5, onde, no ambiente de simulação do Gazebo, a peça consegue ser detectado, mas com um falso positivo. Nas Figuras 6 e 7, podemos ver a detecção de objetos em cima da mesa e dentro de uma impressora 3D. Em ambos os casos, a rede consegue detectar todas as peças sem apresentar nenhum falso positivo ou classificar de forma errada.

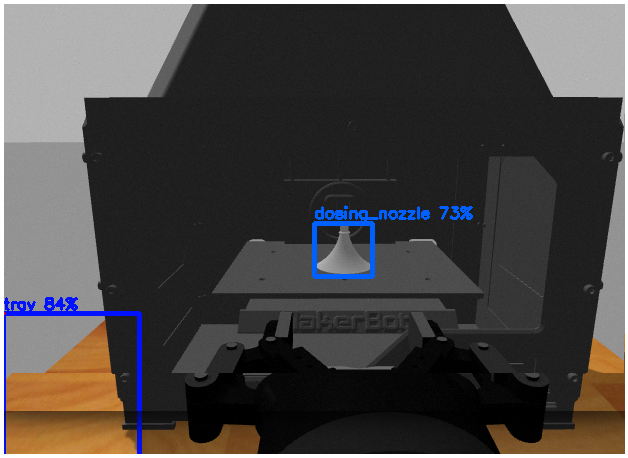

Figura 5. Detecção da peça em ambiente simulado.

Na detecção na nuvem de pontos, podemos ver nas Figuras 8, 9 e 10 uma comparação da imagem original com a peça isolada na nuvem de pontos. Podemos ver que o algoritmo consegue isolar a peça de outros objetos do ambiente, permitindo aplicar o algoritmo de estimação de pose na nuvem de pontos.

Na Tabela 2 temos o Δ, altura e largura das peças estimados, sendo a altura calculada como a diferença da posição no eixo y do ponto superior e o ponto inferior e a largura definida pela diferença na posição no eixo x do ponto na extremidade esquerda e na extremidade direita. Com a altura e largura, podemos saber se é possível pegar a peça considerando largura e altura da garra. Os Δ encontrados definem as peças com as seguintes formas geométricas:

- Marcador: Caixa.

Figura 6. Detecção das peças utilizadas no teste.

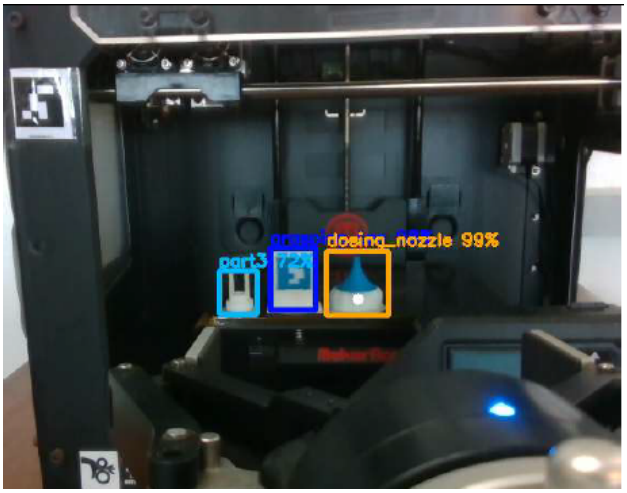

Figura 7. Detecção das peças dentro de uma impressora 3D.

Figura 8. Imagem original vs marcador isolado na nuvem de pontos.

Figura 9. Imagem original vs bico dosador isolado na nuvem de pontos.

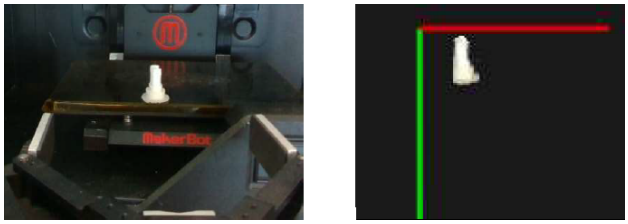

Figura 10. Imagem original vs part3 isolada na nuvem de pontos.

- Bico Dosador: Cilindro.
- Part3: Cilindro.

	Marcador	Bico Dosador	Part3
Δ	0.74	0.29	0.24
Altura[m]	0.042	0.033	0.025
Largura[m]	0.035	0.048	0.022

Tabela 2. Classificação da primitiva geométrica das peças testadas

4.2 Preensão

Para este experimento, o braço robótico UR5, um braço robótico que possui seis graus de liberdade e consegue levantar até 5 kg, retirará a peça de dentro da impressora 3D e colocará a peça em outro local. Para isso ele sair da posição inicial, mover para a direção da impressora, procurar pela peça, pegar a peça, retirar a peça da impressora e colocar a mesma em um local desejado. Como a garra *Robotiq 2F-140* possui um sensor que detecta objetos ao fechar ou abrir a garra, este sensor será utilizado para demonstrar que o objeto foi pego com sucesso. O sensor visual estará instalado próximo da garra, montado no braço, permitindo pegar objetos em qualquer direção, desde que o braço consiga ver a peça. Para este experimento, a orientação da peça será desconsiderada devida as limitações impostas pelo ambiente e da rede de detecção utilizada.

Na Figura 11 temos o gráfico da posição e orientação do braço ao longo do tempo. Ambos foram estimados utilizando Cinemática Direta, aplicada como vista em Mark W. Spong and Vidyasagar (2005), utilizando o sensor de posição das juntas.

Na Figura 12 temos o estado da garra ao longo do tempo, onde cada estado representa:

- 0: Movendo para a posição desejada.
- 1: Objeto detectado enquanto abria os dedos.
- 2: Objeto detectado enquanto fechava os dedos.
- 3: Movimento concluído sem detectar objetos.

Assim, temos que após a amostragem 250 o objeto foi pego pelo braço e foi colocado no destino por volta da amostragem 475.

Por fim, na Figura 13 temos a posição e velocidade das juntas segundo o sensor do UR5. Nele podemos ver que o movimento foi suavizado utilizando Polinômios Quínticos, implementado como visto em Mark W. Spong and Vidyasagar (2005).

O video do experimento pode ser visto em `https://youtu.be/eLoDD7wZDaw`.

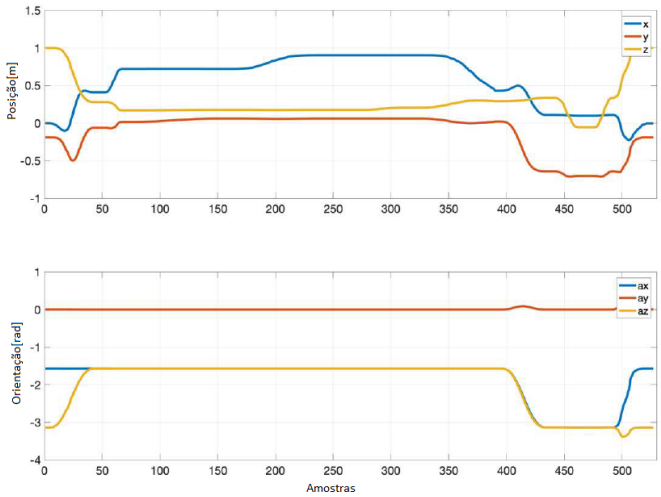

Figura 11. Posição e orientação da garra ao longo do tempo.

Figura 12. Estado da garra ao longo do tempo.

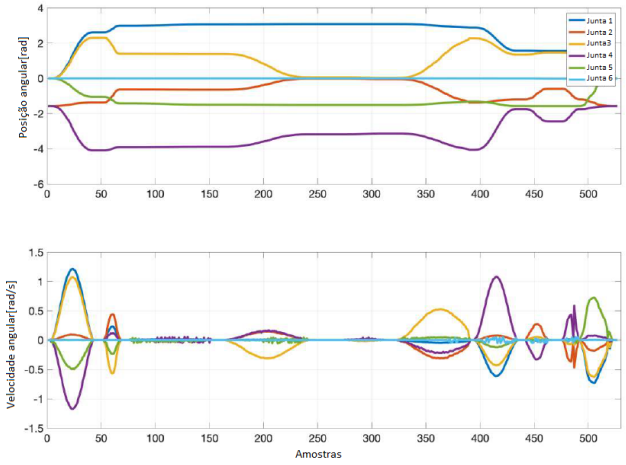

Figura 13. Posição e velocidade das juntas ao longo do tempo.

5. CONCLUSÃO

Neste artigo foi apresentado um algoritmo para pegar objetos utilizando uma rede de aprendizagem profunda em imagem RGB com um algoritmo para estimar a pose em nuvem de pontos. O método apresentado possui a vantagem de utilizar uma rede em imagem RGB, onde possui uma enorme diversidade de redes e base de dados na literatura, e utiliza a nuvem de pontos para fazer uma análise mais profunda e realista da peça. O algoritmo utilizado para converter a *bounding box* da rede é versátil o bastante para poder ser utilizado com outras redes e algoritmos de detecção de objeto em imagem RGB, desde que a nuvem de pontos seja organizada. Para trabalhos futuros, adicionar um algoritmo para pegar a orientação da peça e estudar maneiras de aprimorar o algoritmo de estimação da geometria da peça, pois o mesmo considera que a peça terá uma geometria única, sendo que muitos objetos podem ter mais de uma geometria, como o caso do bico dosador, além de poder possuir formas geométricas muito distintas das três apresentadas.

AGRADECIMENTOS

Este projeto recebeu apoio financeiro do SEPIN/MCTI, no âmbito da 4ª Chamada Coordenada BR-EU no CIT e do programa de pesquisa e inovação European Unions Horizon 2020 no âmbito do Contrato de Doação n° 777096.

REFERÊNCIAS

Coleman, C.A., Narayanan, D., Kang, D., Zhao, T.J., Zhang, J., Nardi, L., Bailis, P., Olukotun, K., Ré, C., and Zaharia, M. (2017). Dawnbench : An end-to-end deep learning benchmark and competition.

Girshick, R. (2015). Fast r-cnn. 1440–1448. doi:10.1109/ICCV.2015.169.

Guo, J., He, H., He, T., Lausen, L., Li, M., Lin, H., Shi, X., Wang, C., Xie, J., Zha, S., Zhang, A., Zhang, H., Zhang, Z., Zhang, Z., Zheng, S., and Zhu, Y. (2020). Gluoncv and gluonnlp: Deep learning in computer vision and natural language processing. *Journal of Machine Learning Research*, 21(23), 1–7. URL `http://jmlr.org/papers/v21/19-429.html`.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. doi:10.1109/CVPR.2016.90.

Jain, S. and Argall, B. (2016). Grasp detection for assistive robotic manipulation. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2015–2021. doi:10.1109/ICRA.2016.7487348.

Lemos, C., Farias, P., Simas Filho, E., and Scolari Conceicao, A. (2019). Convolutional neural network based object detection for additive manufacturing. In *2019 19th International Conference on Advanced Robotics (ICAR)*, 420–425. doi:10.1109/ICAR46387.2019.8981618.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S.E., Fu, C.Y., and Berg, A.C. (2016). Ssd: Single shot multibox detector. In *ECCV*.

Mark W. Spong, S.H. and Vidyasagar, M. (2005). *Robot Modeling and Control*. John Wiley & Sons, 1ª edition.

Radu Bogdan Rusu, Blodow, N., Marton, Z., Soos, A., and Beetz, M. (2007). Towards 3d object maps for autonomous household robots. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3191–3198. doi:10.1109/IROS.2007.4399309.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779–788.

Rusu, R.B. and Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China.

Zapata-Impata, B.S. (2017). Using geometry to detect grasping points on 3d unknown point cloud. In *ICINCO*.