Preensão de objetos em 6D Utilizando Redes Neurais Convolucionais e Curvaturas

Daniel M. de Oliveira * **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 e andre.gustavo@ufba.br).*

Abstract: In this work, a 6D grasping system using convolutional neural networks and point cloud will be presented. Detection of the object and estimation of its pose in 6D is done in an RGB image using a convolutional neural network, and the point cloud is used to estimate the best 6D pose to hold the object considering the curvatures on its side. The validation of the proposed system was carried out in a gazebo simulation environment, using the 3D models of the objects made available by the database used in the neural network, the UR5 robotic arm and an RGB-D sensor.

Resumo: Neste trabalho será apresentado um sistema de preensão em 6D utilizando redes neurais convolucionais e nuvem de pontos. A detecção do objeto e estimação da sua pose em 6D é feita em imagem RGB utilizando uma rede neural convolucional, e a nuvem de pontos é utilizada para estimar a melhor pose 6D para realizar a preensão do objeto considerando as curvaturas na lateral do mesmo. A validação do sistema proposto foi feita em ambiente de simulação gazebo, utilizando os modelos 3D dos objetos disponibilizados pela base de dados utilizada na rede neural, o braço robótico UR5 e um sensor RGB-D.

Keywords: Redes Neurais Convolucionais; Aprendizado Profundo; Nuvem de Pontos; Preensão; Manipulador Robótico; Visão Computacional.

Palavras-chaves: Convolucional Neural Networks; Deep Learning; Point Cloud; Grasping; Robotic Manipulators; Computer Vision.

1. INTRODUÇÃO

Com a evolução das tecnologias e o avanço da indústria 4.0, tarefas autônomas tem se tornado cada vez mais comum em ambiente industrial. Na robótica, diversas tarefas como navegação, mapeamento de ambientes, preensão de objetos e fabricação de peças tem se tornado cada vez mais autônomas, retirando quase que totalmente a necessidade de um operador. Com a evolução dos *hardware* e, consequentemente, das redes de aprendizado profundo, tarefas como detecção e localização de objetos podem facilmente ser realizadas utilizando estes tipos de redes, algumas redes como *YOLO* (Padmanabula et al. (2020)) e *SSD* (Liu et al. (2016)) são comumente utilizadas para detecção de objetos em imagens RGB, enquanto redes como *PoseCNN* (Xiang et al. (2018)) e *DenseFusion* (Wang et al. (2019)) são utilizadas para estimar a localização de um objeto em 6D, ou seja, considerando a posição *(x,y,z)* e orientação *(ox,oy,oz)*. Alguns trabalhos, como Viturino et al. (2020), utilizam duas redes neurais para realizar a tarefa de preensão, uma para o processo de identificação do objeto desejado e outra para o processo de seleção de região para realizar a preensão. Além das redes neurais, trabalhos em imagem RGB-D, utilizando nuvem de pontos, também tem evoluído bastante graças a redução de custo dos sensores. Trabalhos como Jain and Argall (2016) e Zapata-Impata et al. (2019) são exemplos disto, onde a nuvem de pontos é utilizada para analisar um objeto e estimar a melhor região para pegar o mesmo.

Em trabalho publicado anteriormente, Oliveira et al. (2020), a rede neural convolucional *SSD* foi utilizada em conjunto de estimação de primitivas geométricas para realizar a preensão de objetos dentro de uma impressora 3D, mas, este trabalho possuía alguns problemas como desconsiderar a orientação da peça e considerava que o objeto sempre possuía uma geometria única, enquanto geralmente possuem uma geometria complexa.

Tendo os problemas encontrados anteriormente, este trabalho apresentará um sistema de preensão em 6D, onde uma rede neural convolucional realizará o processo de detecção e estimação de pose 6D dos objetos enquanto um algoritmo de preensão analisará as curvaturas na lateral do objeto para buscar o melhor local para se realizar a preensão. O sistema foi desenvolvido utilizando o *ROS* (Stanford Artificial Intelligence Laboratory et al.), *PyTorch* (Paszke et al. (2019)) e *PCL* (Rusu and Cousins (2011)). O sistema final foi validado no ambiente de simulação *Gazebo* utilizando o braço robótico *UR5*, da *Universal Robots*, e uma câmera RGB-D.

Este trabalho está organizado da seguinte maneira: A seção 2 apresenta a rede neural convolucional utilizada; a seção 3 apresenta o algoritmo de preensão desenvolvido; a seção 4 apresenta os resultados dos experimentos reali-

zados; a seção 5 apresenta a conclusão do trabalho além de apresentar possíveis trabalhos futuros.

2. ESTIMAÇÃO DE POSE EM 6D BASEADA EM SEGMENTAÇÃO

A Rede neural utilizada neste trabalho para detectar objetos e estimar pose 6D foi a *Estimação de Pose em 6D Baseada em Segmentação* (Hu et al. (2018)), pelo fato da rede ser treinada para dois conjuntos de dados diferentes: *YCB Dataset* (Xiang et al. (2018)) e *Linemod Dataset* Hinterstoisser et al. (2012), permitindo testar o sistema em mais objetos, além de possuir a detecção dos mesmos embutido, facilitando o processo de prototipagem e testes do produto final.

A rede funciona executando duas ações em simultâneo: segmentação de objetos e regressão 2D de pontos chaves do objeto, como podem ser vistos na imagem 1. Após esse processo, *PnP* é aplicado para estimar a pose final do objeto.

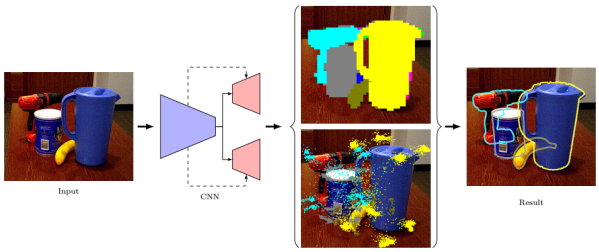

Figura 1. Funcionamento da rede utilizada (Hu et al. (2018)).

A arquitetura da rede neural convolucional utilizada possui dois decodificadores com um codificador em comum, como visto na figura 1. O codificador utilizado foi a rede *Darknet-53*, da arquitetura da rede *YOLOv3* Padmanabula et al. (2020). Para os decodificadores, um deles realizará o processo de segmentação dos objetos, enquanto o outro realizará o processo de regressão 2D de pontos chaves do objeto, como visto na imagem 2.

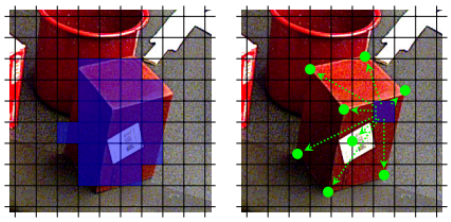

Figura 2. Resultado das redes de decodificação utilizadas (Hu et al. (2018)).

Para realizar o processo de estimação de pose 6D, primeiro considere um número N de pontos chaves 3D, onde os pontos 3D são equivalentes aos pontos 2D encontrados. Considerando que os conjuntos de dados disponibilizam os modelos 3D dos objetos dos mesmos, é feita uma correspondência entre os pontos encontrados na imagem RGB e os pontos encontrados nos modelos 3D. Com isso, são utilizados n=*10* pontos de melhor confiabilidade, onde este valor de n foi escolhido por possuir melhor equilíbrio entre tempo de execução e qualidade da detecção. Por fim, ePnP Lepetit et al. (2009) é utilizado para realizar a estimação da pose final.

Para os resultados, a métrica utilizada foi a *ADD*, onde ela considera a pose de um objeto como correta caso ela tenha um erro menor do que 10% do diâmetro do objeto. Para os experimentos realizados por Hu et al. (2018), foi utilizado uma resolução de *608x608* e o treinamento para os conjuntos de dados foram feitos com as seguintes épocas: 300 para o *Linemod* e 30 para o *YCB*. Com isso, foi possível obter um *ADD* médio de 27.0 no *Linemod Dataset* e 39.0 no *YCB Dataset*, superando outras redes do estado da arte como PoseCNN(Xiang et al. (2018)).

3. PREENSÃO BASEADA EM CURVATURAS

Para o algoritmo de preensão utilizado, inicialmente, considera uma nuvem de pontos p contendo somente o objeto que desejamos pegar. Como a detecção do objeto e da pose 6D é feita em imagem RGB, assim, com o conhecimento do objeto, da pose e tendo o modelo 3D, podemos criar uma nuvem de pontos na mesma pose estimada e com o objeto detectado pela rede neural. O algoritmo foi feito baseado nos trabalhos de Jain and Argall (2016), Satish et al. (2019) e Zapata-Impata et al. (2019), onde são geradas possíveis regiões de preensão e utilizar análise de curvaturas para detectar a melhor curvatura para se realizar a preensão.

3.1 Geração de Regiões de Preensão

Para gerar as possíveis regiões de preensão, considere h_g como a altura de cada região e H_o a altura do objeto definido pela nuvem de pontos p, sendo que a altura do objeto pode ser calculada na nuvem de pontos utilizando a seguinte fórmula, onde y_{max} e y_{min} são os maiores e menores valores no eixo y respectivamente, onde o eixo y representa o eixo vertical e o eixo x o eixo horizontal com relação à câmera:

$$H_o = y_{max} - y_{min} \tag{1}$$

Com isso podemos quebrar o objeto em K pedaços, onde K é definido por:

$$K = \frac{H_o}{h_g} \tag{2}$$

Diferente de como foi feito em Zapata-Impata et al. (2019), onde a região de análise do objeto foi limitada ao redor do centro da mesma, analisaremos o objeto todo, por isso são geradas K peças seguindo o algoritmo abaixo, onde $fp(p, y_{min}, y_{max})$ indica o uso de filtro passa faixa para cortar a peça p no eixo y e j_i indica a região gerada:

Algoritmo de Geração de Regiões de Preensão

$y \leftarrow y_{min}$
$y_f \leftarrow y_{min} + h_g$
while $i < K$ **do**
  $j_i \leftarrow fp(p, y, y_f)$
  $y \leftarrow y_f$
  $y_f \leftarrow y_f + h_g$
  $i \leftarrow i + 1$

end while

Assim podemos transformar um objeto como visto na figura 3 em vários objetos menores, como visto na figura 4.

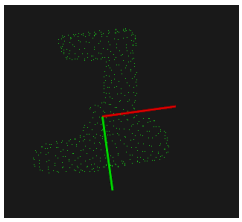

Figura 3. Objeto a ser analisado.

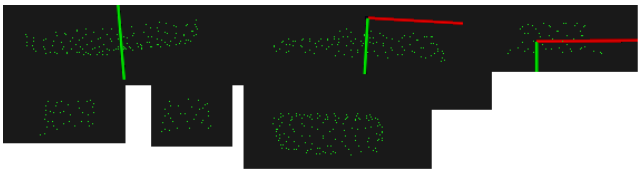

Figura 4. Objeto separado em várias possíveis regiões de preensão.

3.2 Calculo da Curvatura

Para o cálculo de curvatura, utilizaremos uma abordagem igual à vista em Jain and Argall (2016). Assim, considere que cada região de preensão possui um número de pontos N_i, assim, podemos calcular a *centroid* dos pontos contidos N_i com:

$$p_c = \frac{\sum_{i=1}^{N_i} j_{oi}}{N_i}, j_{oi} \in j_i \tag{3}$$

Com isso, podemos definir a matriz de covariância como:

$$C = \frac{1}{N_i} \sum_{i=1}^{N_i} \varepsilon_i \cdot (j_{oi} - p_c) \cdot (j_{oi} - p_c)^T \tag{4}$$

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:

$$C \cdot \boldsymbol{v}_t = \lambda_t \cdot \boldsymbol{v}_t, t \in \{1, 2, 3\} \tag{5}$$

Usando *PCA(Principal Component Analysis)*, podemos encontrar a variação da superfície σ_{j_*} em volta de cada ponto j_* como:

$$\sigma_{j_*} = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2} \tag{6}$$

3.3 Detecção de Geometria Simples

Para detectar se o objeto possui uma geometria simples, calcularemos a curvatura de cada região de preensão como visto na seção 3.2. Assim, podemos definir uma métrica Δ, utilizada para definir a primitiva geométrica da região, como visto em Jain and Argall (2016):

$$\Delta = \frac{M^*}{M} \tag{7}$$

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

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

Após verificar a geometria de cada região, verificaremos se todas possuem a mesma primitiva e a mesma largura dentro de uma margem de erro e. Esta margem de erro foi adicionada pelo fato de a geometria dos objetos não serem perfeitas além de possíveis erros de resolução no processo de conversão e *downsample* do modelo 3D para nuvem de pontos. Caso o objeto seja uma primitiva geométrica, realizaremos a preensão no centro da peça, pelo fato da mesma ser uniforme.

3.4 Seleção de Melhor Região

Caso o objeto não seja uma primitiva geométrica, adotaremos uma abordagem similar à vista em Zapata-Impata et al. (2019), onde calcularemos a curvatura nas extremidades laterais de cada região, pois, as mesmas entrarão em contato com a garra do braço robótico, e, quanto menor a curvatura da região, mais linear ela será, assim, produzindo uma preensão mais segura. Assim, considerando x_{maxci} e x_{minci} as extremidades de cada região, calcularemos as curvaturas $curvmax_i$ e $curvmin_i$ de M pontos ao redor destas extremidades. Com isso, encontraremos a pontuação da curvatura da região como:

$$P_i = curvmax_i + curvmin_i \tag{8}$$

Assim, considerando a abertura da garra como W_g, e a largura da região de preensão como:

$$L_i = x_{maxci} - x_{minci} \tag{9}$$

Selecionaremos a região que possua o menor valor de P_i e possuir $L_i < W_g$.

4. RESULTADOS

Os resultados deste artigo serão divididos em duas partes: A primeira, vista na seção 4.1, focará no algoritmo de preensão proposto e estimação de onde melhor pegar um objeto, utilizando alguns objetos do *YCB dataset*; e a segunda, vista na seção 4.2, será a execução do sistema completo em tarefas de preensão no ambiente de simulação Gazebo.

4.1 Geração de Regiões de Preensão

Para gerar os objetos em nuvem de pontos, pegaremos os arquivos no formato *.obj* disponibilizados pelo conjunto de dados. Com os arquivos, utilizaremos *Poisson Disk*

Sample (Yuksel (2015)) para reduzir o número de pontos do modelo para 256 e converter para uma nuvem de pontos.

Nas figuras 5 e 6 temos a estimação da melhor região para pegar os objetos. A região indicada em vermelho é a região indicada para se realizar a preensão por apresentar menor curvatura nas laterais. Já na figura 7 temos um objeto que possui a geometria que pode ser aproximada a uma primitiva geométrica, neste caso um cilindro, por isso, o local indicado para se pegar esta categoria de objeto seria no centro (Jain and Argall (2016)).

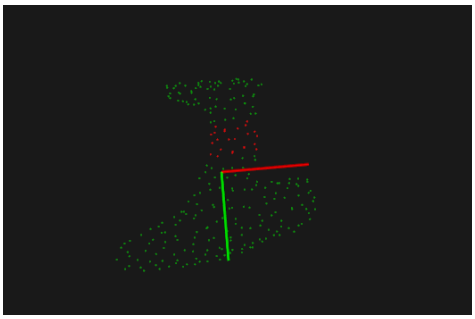

Figura 5. Melhor região para pegar a furadeira.

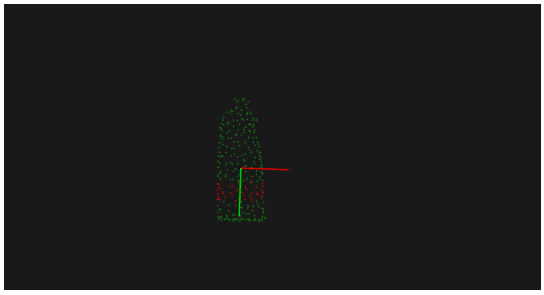

Figura 6. Melhor região para pegar a garrafa de detergente.

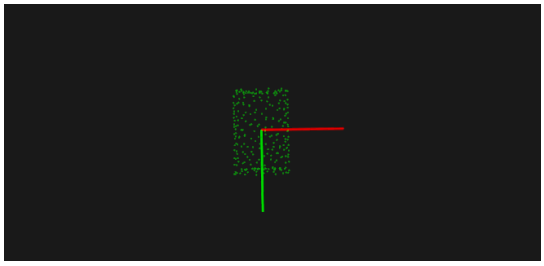

Figura 7. Melhor região para pegar a lata de extrato de tomate.

Um dos problemas encontrados foi no cálculo de curvaturas em regiões de baixa largura com relação a outras regiões de preensão do objeto quando a altura de cada região h_g é pequena, fazendo com que a região acabe tendo um valor baixo na curvatura mesmo quando a mesma possui visivelmente uma curvatura alta.

4.2 Preensão em 6D

Para realizar os experimentos com o sistema completo, utilizaremos o ambiente de simulação *gazebo* e o *ROS*. Os objetos escolhidos para este experimento foram os quais possuíam um modelo 3D sem defeitos na sua geometria e foi possível estimar uma estimação de pose satisfatória em ambiente de simulação. O sistema completo funcionará da seguinte maneira:

- Detectar o objeto e a pose utilizando a rede neural vista na seção 2.
- Com a pose e o objeto conhecido, é gerado o objeto na nuvem de pontos com a mesma pose vista com relação a câmera.
- Aplica o algoritmo de preensão visto na seção 3.
- Utilizar transformada homogênea (Briot (2015)) para estimar a pose com relação ao braço para mover o braço para a posição de preensão da peça.

Para realizar os experimentos, utilizaremos um *UR5* da *Universal Robots* em conjunto de um sensor RGB-D. Na figura 8 temos o manipulador robótico utilizado, na pose inicial utilizada em todos os experimentos, onde o cubo a cima dele representa o sensor RGB-D utilizado, sendo que o mesmo está fixo na última junta do braço. A pose inicial do braço para todos os experimentos foram *x = -0.4m, y=-0.01m, z=0.5m, ox=3.14 rad, oy=0 rad e oz=1.57 rad*. Os experimentos foram realizados no Ubuntu 18 com processador *AMD Ryzen 5 3600* e placa gráfica *Nvidia RTX 2060*. O funcionamento do sistema pode ser visto em `https://youtu.be/P67bEibGE1U`.

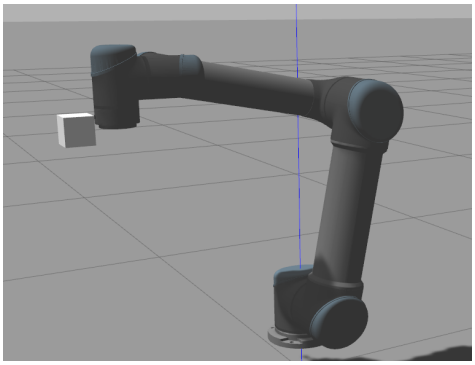

Figura 8. Manipulador robótico utilizado nos experimentos.

O primeiro experimento realizado foi utilizado uma lata de extrato de tomate, onde a mesma se encontra na pose *x = -0.48m, y=-0.07m, z=0.4m, ox=3.14 rad, oy=0 rad e oz=1.57 rad*. Na figura 9 temos a posição das juntas temporalmente, onde podemos ver que o movimento foi suavizado utilizando polinômios quínticos. Nas figuras 10, 11 e 12 temos a posição e orientação temporalmente. A pose final do braço foi *x = -0.4871m, y=-0.0812m, z=0.43m, ox=3.10 rad, oy=0.04 rad e oz=1.63 rad*, sendo assim, possuindo um erro em cada eixo de *x = 1.47%, y=16%, z=7.5%, ox=1.2%, oy=4% rad e oz=3.8%*, onde temos boa parte dos erros no esperado pela métrica *ADD*.

Para o próximo experimento, foi utilizado uma garrafa de detergente com a pose de *x = -0.5m, y=0.02m, z=0.3m, ox=-2.5 rad, oy=0 rad e oz=0.7 rad* com relação ao braço. Na figura 13 temos a posição das juntas temporalmente, onde, assim como no experimento anterior, podemos ver a suavização do movimento das juntas. Nas figuras 14, 15 e 16 temos a posição e orientação ao longo do tempo, onde podemos ver que a pose final do braço foi de *x = -0.511m, y=0.018m, z=0.34m, ox=-2.72 rad, oy=0.023 rad e oz=0.714 rad*, tendo assim um erro em cada eixo de *x =*

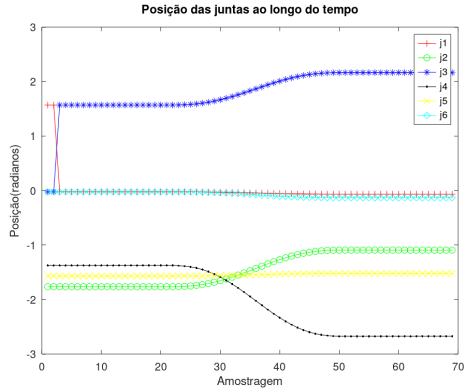

Figura 9. Posição das juntas ao longo do tempo na tarefa de pegar a lata de extrato de tomate.

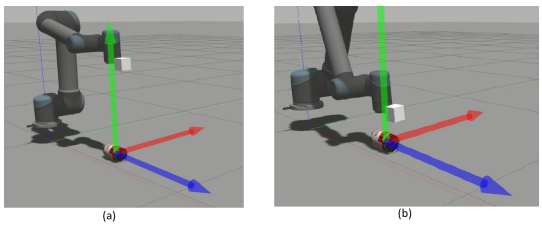

Figura 10. Pose do braço ao longo do tempo na tarefa de pegar a lata de tomate, onde: (a) é uma pose intermediária e (b) é a pose final.

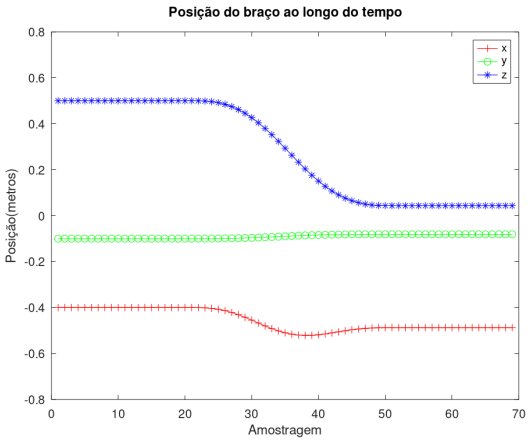

Figura 11. Posição do braço ao longo do tempo na tarefa de pegar a lata de extrato de tomate.

2.2%, y=10%, z=13%, ox=8.8%, oy=2.3% rad e oz=2% rad, sendo assim, ficando similar ao experimento anterior, onde quase todos os eixos possuem erro dentro do *ADD* esperado.

Em questão de desempenho, obtivemos um tempo de execução médio em *gpu* para a rede neural de 0.045 segundos e o algoritmo de preensão em *cpu* de 0.001 segundos, fazendo com que o sistema possa ser utilizado em tarefas que necessitem de baixo tempo de execução.

5. CONCLUSÃO

Neste artigo foi apresentado um sistema de preensão de objetos em 6D, onde foi utilizado uma rede neural convolucional para realizara o processo de detecção e estimação

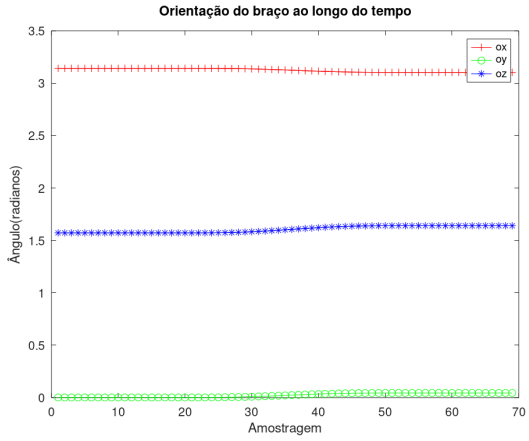

Figura 12. Orientação do braço ao longo do tempo na tarefa de pegar a lata de extrato de tomate.

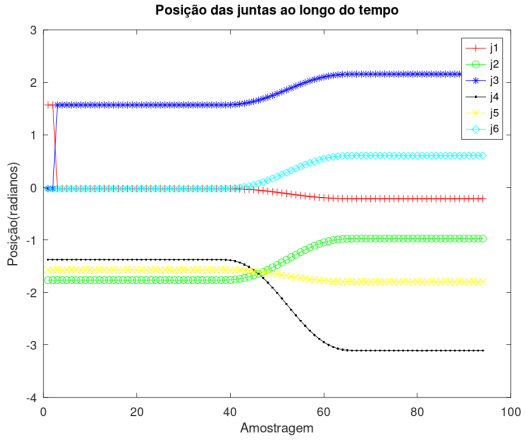

Figura 13. Posição das juntas ao longo do tempo na tarefa de pegar a garrafa de detergente.

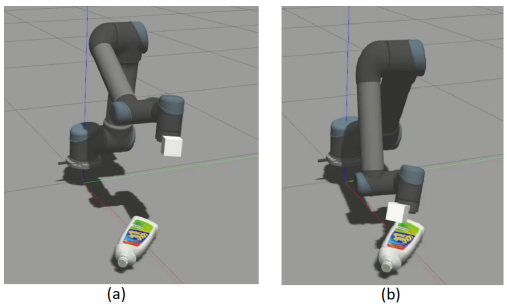

Figura 14. Pose do braço ao longo do tempo na tarefa de pegar a garrafa de detergente, onde: (a) é uma pose intermediária e (b) é a pose final.

de pose, enquanto o algoritmo de preensão utilizado foi de autorial própria. O sistema apresentado possui a vantagem de poder realizar a preensão de objetos considerando a pose dele além de considerar um possível melhor local para se pegar o objeto, sem a necessidade de conhecer o objeto previamente, sendo somente necessário possuir o objeto isolado na nuvem de pontos. Além disso, o algoritmo de preensão proposto é independente da rede neural convolucional, podendo ser utilizado em diversos sistemas, com ou sem estimação de pose, além de poder ser utilizado

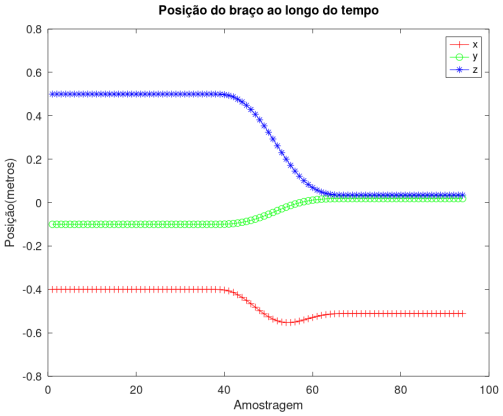

Figura 15. Posição do braço ao longo do tempo na tarefa de pegar a garrafa de detergente.

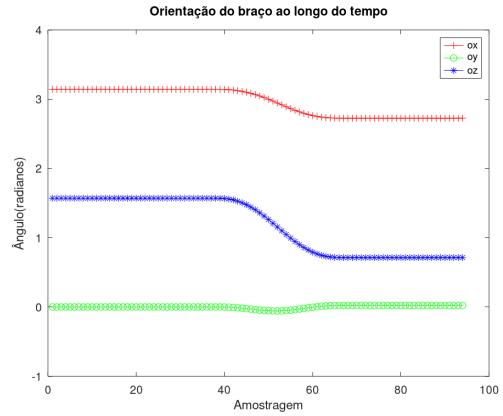

Figura 16. Orientação do braço ao longo do tempo na tarefa de pegar a garrafa de detergente.

com outras redes neurais convolucionais mais precisas que podem surgir no futuro. Para trabalhos futuros, o sistema será validado em um número maior de objetos, aprimorar o algoritmo de preensão proposto baseado em problemas encontrados neste número maior de objetos e, por fim, validar o sistema no braço robótico real.

AGRADECIMENTOS

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001 e pelo CNPQ termo de outorga numero 311029/2020-5.

REFERÊNCIAS

Briot, S.; Khalil, W. (2015). *Dynamics of Parallel Robots: From Rigid Bodies to Flexible Elements*. Springer International Publishing, 1ª edition.

Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., and Navab, N. (2012). Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. volume 7724. doi:10.1007/978-3-642-33885-4_60.

Hu, Y., Hugonot, J., Fua, P., and Salzmann, M. (2018). Segmentation-driven 6d object pose estimation.

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.

Lepetit, V., Moreno-Noguer, F., and Fua, P. (2009). Epnp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision*, 81. doi:10.1007/s11263-008-0152-6.

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*.

Oliveira, D., Lemos, C., and Scolari Conceicao, A. (2020). Sistema de preensão robótica utilizando redes neurais convolucionais e primitivas geométricas. doi:10.48011/asba.v2i1.1097.

Padmanabula, S., Puvvada, R., Sistla, V., and Kolli, V.K.K. (2020). Object detection using stacked yolov3. *Ingénierie des systèmes d information*, 25, 691–697. doi:10.18280/isi.250517.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, 8024–8035. Curran Associates, Inc.

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.

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.

Satish, V., Mahler, J., and Goldberg, K. (2019). On-policy dataset synthesis for learning robot grasping policies using fully convolutional deep networks. *IEEE Robotics and Automation Letters*.

Stanford Artificial Intelligence Laboratory et al. (????). Robotic operating system. URL `https://www.ros.org`.

Viturino, C., Filho, K., Oliveira, D., Lemos, C., and Scolari Conceicao, A. (2020). Redes neurais convolucionais para identificação e preensão robótica de objetos. doi:10.48011/asba.v2i1.1163.

Wang, C., Xu, D., Zhu, Y., Martín-Martín, R., Lu, C., Fei-Fei, L., and Savarese, S. (2019). Densefusion: 6d object pose estimation by iterative dense fusion. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 3338–3347.

Xiang, Y., Schmidt, T., Narayanan, V., and Fox, D. (2018). Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes.

Yuksel, C. (2015). Sample elimination for generating poisson disk sample sets. *Computer Graphics Forum*, 34. doi:10.1111/cgf.12538.

Zapata-Impata, B., Gil, P., Pomares, J., and Medina, F. (2019). Fast geometry-based computation of grasping points on three-dimensional point clouds. *International Journal of Advanced Robotic Systems*, 16. doi:10.1177/1729881419831846.