2022 IEEE 18th International Conference on Automation
Science and Engineering (CASE)
August 20-24, 2022. Mexico City, Mexico

2022 IEEE 18th International Conference on Automation Science and Engineering (CASE) | 978-1-6654-9042-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/CASE49997.2022.9926580

Augmenting Vision-Based Grasp Plans for Soft Robotic Grippers using Reinforcement Learning

Vighnesh Vatsal[1] and Nijil George[1]

***Abstract*— Vision-based techniques for grasp planning of robotic end-effectors have been successfully deployed in pick-and-place tasks. However, for computing the optimal grasp, they assume the gripper to be of a rigid parallel-jaw type or a single-point vacuum suction-based design. Planners for soft robotic grippers have used learning from demonstration or heuristics that rely on the compliance of the gripper to achieve a grasp. We demonstrate a model-free reinforcement learning (RL) approach that modifies vision-based grasp plans generated for parallel-jaw grippers and adapts them to grasping with a four-fingered soft gripper. The observed state of the RL model is comprised of the grasp plans from the vision module, the deformation of the fingers, and the pose of the end-effector. The RL model controls each finger separately, discovering grasp synergies during training. This approach is compared to a baseline grasp synergy where all four fingers simultaneously enclose the object. In simulation, we achieve a pick-and-place success rate of 58.4% with the RL model for top-down grasping, compared to 43.2% with the baseline grasp synergy.**

I. Introduction

Grasping is a fundamental capability for robotic systems to be effectively deployed in real-world scenarios. Planning a grasp for a given robotic end-effector involves reasoning about the geometries of both the gripper and the target object [1]. Therefore grasp planning is typically formulated as a vision problem, where given certain assumptions on the structure and physical properties of the gripper and target object, the goal is to determine the optimal pose of the gripper with respect to the object based on an RGB or RGB-D image.

Present approaches to robotic grasp planning being widely used in practice rely on vision-based techniques such as convolutional neural network (CNN) models. These models typically take a single-view or multi-view image of the target object and generate the optimal gripper pose as the output, based on a grasp quality metric. Some of the key assumptions in these techniques are that the grippers have rigid contact with the target object, being either of a parallel-jaw or two-fingered design, or based on a single point of contact through vacuum suction. Additionally, the target objects are assumed to be non-deformable under typical grasping forces.

With recent advances in soft robotics, compliant, underactuated, and deformable grippers are witnessing wider adoption. Compared to rigid designs, soft grippers provide the advantages of enhanced safety in collaborative usage scenarios, and adaptability to a larger range of target objects due to compliance. However, they also present challenges in planning effective grasps due to difficulties in modeling the contact with the target object and deformations of the fingers. As a result, grasp plans for these types of grippers have relied on heuristics which assume that an approximately correct plan will succeed due to compliance, or by using reinforcement learning (RL) models trained directly on physical systems.

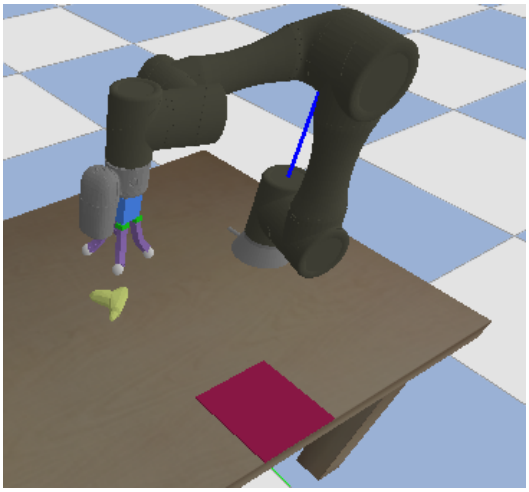


Fig. 1: The soft grasp planning simulation environment. The objective is to pick the target object from the top and place it in the red region on the table.

We aim to develop grasp plans for soft robotic grippers in simulation, allowing for greater exploration of grasp strategies compared to direct training on physical systems. Concurrently, it was found that the antipodal grasps generated by CNN models can serve as a good starting point for an RL agent to control the four-fingered soft gripper considered here (Fig. 1). The grasp planning method proposed in this paper can be summarized as follows:

- Implementation of a Generative Grasping CNN (GG-CNN) [2] in a PyBullet environment [3] for a table-top pick-and-place task with a 6-degrees of freedom robotic arm, generating an optimal pixel-wise grasp based on an RGB image for a top-down pick.
- Using the robot wrist pose and grip width from the GG-CNN, along with computation of the 3D curvature and torsion of each of the fingers as input observations for an RL agent.
- Training a model-free deep RL agent using the Soft Actor-Critic (SAC) algorithm to generate optimal wrist poses and finger bending commands based on the above

[1]The authors are with TCS Research & Innovation, Tata Consultancy Services, Bengaluru, Karnataka - 560066, India. e-mail: vighnesh.vatsal@tcs.com, george.nijil@tcs.com

observations.

The training and testing of the RL agent was performed on the dataset of randomly-generated 3D objects from [4]. The RL agent was compared to a baseline grasp synergy where each of the four fingers open and close together, attempting to envelop the object. The RL agent had a grasp success rate of 58.4% on the test set, compared to 43.2% with the baseline strategy.

II. Related Work

A. Soft Robotic Grippers

Soft robotic grippers are grippers made up of deformable materials or flexible structures. They are underactuated in the sense that they perform motions with large degrees of freedom while acted upon by a limited number of control variables. Soft grippers are normally classified based on the type of drive modes - fluid drive, cable drive, etc. A more detailed review of the various drive modes can be found in [5]. Following are some of the soft robotic hands in existing literature. Pisa/IIT SoftHand, presented in [6] is a cable driven soft gripper. RBO hand 2 introduced in [7] is a pneumatically driven anthropomorphic robotic gripper which is capable of performing grasps similar to a human hand. DRL Softhand, described in [8] is pneumatically driven and has force and bend sensors embedded into it. [9] presents a pneumatically actuated dexterous soft gripper and demonstrates in-hand manipulation, which forms the basis for this work.

B. Learning-based Grasp Planning

As robot manipulators advance to be used in unstructured environments, the type of objects they have to grasp also diversify. Traditional rigid grippers and control schemes proved to be insufficient for all use cases. This has led to increased research towards soft grippers and their learning-based control. A reinforcement learning (RL) approach is used for designing a controller for a soft gripper in [10], where the training data is from human demonstrations and the training of the RL agent is done directly on hardware. An end-to-end deep learning (DL) approach for generating grasps for a soft gripper is proposed in [11], where the input video stream is converted by a convolutional neural network (CNN) to discrete position outputs for the gripper and wrist, and training is done on labeled data. In [8] the authors propose an approach for grasp planning of a soft gripper equipped with proprioceptive sensors—force and bend, with the data from these sensors used to generate a grasp and identify the object being grasped. There have been attempts at solving the grasping problem for a two-fingered rigid gripper using a combination of DL (vision) and RL in [4] and [2], while [12] extended this approach for improved performance by incorporating multi-view images. [13] proposed an end-to-end training approach for a combination of DL (vision) and RL agent for grasping and other tasks using a rigid gripper, where the RL agent is based on guided policy search [14], [15]. However, these works cannot be directly applied to the grasp planning of soft grippers because of its underactuated nature. [16] suggested a mathematical way to find the effective grasp direction for a soft gripper, which is then aligned with the grasp closure direction predicted by the DL (vision) network. However, this method cannot be used for all type of soft grippers and hand configurations.

III. System Description

The proposed grasping system is composed of a Techman TM5-700 robotic arm as the manipulator with a four-fingered soft gripper attached as the end-effector. An RGB camera is affixed to the wrist of the manipulator for perception. The visual input from the camera is given as input to a GG-CNN [2] module. The output of the GG-CNN, described in the next section, captures the details essential for a successful grasp for a rigid two fingered gripper. These vectors are passed as part of the environment state to an RL agent. In addition to the grasp vectors, the environment state also comprises of additional information in the form of curvature of the soft fingers and the position of the wrist. Using these observations, the RL agent generates the required control signals for the four fingers of the soft gripper, as well as the angle and height of the wrist. In the subsequent subsections each of the components are explained in detail. The simulation environment used for this work is PyBullet [3], with the robotic arm design and baseline code for grasping using RL obtained from [17].

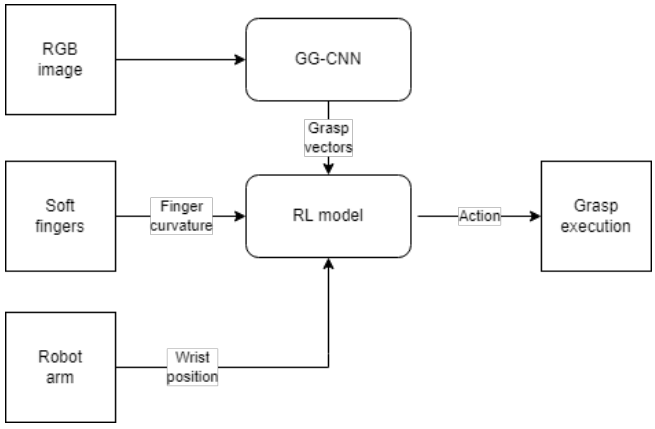
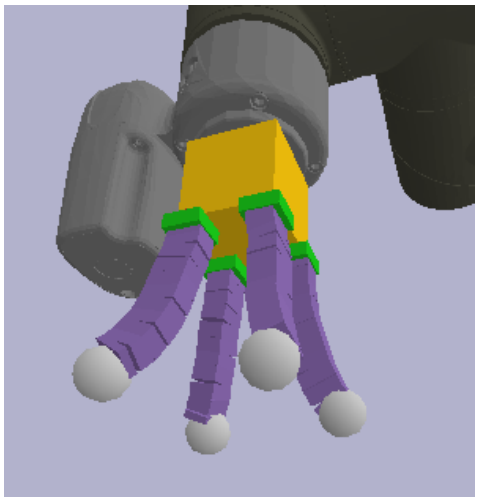


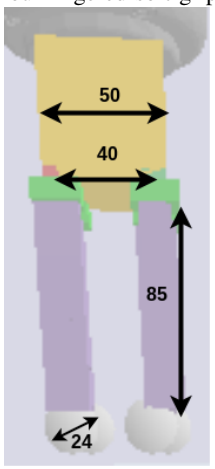
Fig. 2: Overview of the proposed approach.

A. Soft Gripper

The soft gripper used in this work was configured as a combination of four flexible fingers attached to a base as shown in Fig. 3a. Each individual finger can be controlled independently by a scalar input. The finger models are obtained from [9], with their dynamics being described in [18]. The related codebase can be found here [19]. The CAD model of each finger is composed of ten segments, approximating a soft and flexible structure, with a spherical tip at the end. The length and dimensions of the resulting gripper is shown in Fig. 3b. This type of design is inspired from existing commercial grippers such as the four-finger configuration of the mGrip modular gripping system from Soft Robotics Inc. [20]. This design enables the fingers to



(a) Four-fingered soft gripper.



(b) The fingers are mounted on a cubical box attached to the robot's wrist, at its vertices in a symmetric manner. Dimensions in mm.

Fig. 3: Four-fingered gripper designed using the soft robotic fingers described in [9].

enclose a large variety of objects that can be encountered in application scenarios such as retail and grocery stores.

In terms of control, each finger is composed of a series of revolute joints connecting successive segments, approximating the effect of pneumatic actuation. A single scalar command is given to each finger, with the same command applied to all the segments, resulting in a quadratic curve being traced out by the finger.

However, when the fingers encounter the target object, they are bent and distorted according to the contact dynamics parameters, including the friction and restitution coefficients described in [18] (here 0.8 and 0.7 respectively for each segment). Each of the segments bends and twists from the default start state. This interaction between the fingers and the target object is captured by the geometric quantities of curvature (κ) and torsion (τ) in 3D space:

$$\kappa = \frac{|v \times a|}{|v|^3} \tag{1}$$

$$\tau = \frac{-(v \times a) \cdot \dot{a}}{|v \times a|^2} \tag{2}$$

These quantities are computed in a piece-wise manner for each segment of the fingers (κ_i, τ_i for the i^{th} segment). Considering a sequence of position vectors p_i of the i^{th} segment in 3D, v and a are, respectively, the first and second numerical derivatives, computed using a central difference gradient rule:

$$v_i = \frac{p_{i+1} - p_{i-1}}{2\gamma} \tag{3}$$

$$a_i = \frac{v_{i+1} - v_{i-1}}{2\gamma} \tag{4}$$

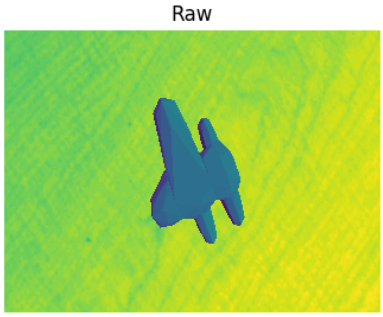
The homogeneous step size γ can be arbitrary, as it cancels out in Eq. (1) and Eq. (2). The quantities $\{\kappa_i, \tau_i\}$ are included in the observed state of the environment, based on which the RL agent decides the appropriate control action. They provide the agent with a notion of how the fingers distort when coming in contact with an object. While these quantities can be determined to an arbitrary degree of accuracy in simulation, they need to be estimated during physical experiments. Various techniques can be employed for this purpose, such as offline dynamics models built using data from visual marker-based nonlinear system identification trials [21]; or using emerging techniques in embedding curvature sensors within soft robots [22]. Estimation of finger curvature and torsion can be implemented on a physical system through a combination of off-board cameras, pressure sensors for pneumatic actuators, encoder readings for cable-driven fingers, and embedded flexible sensors.

IV. Methods

As shown in Fig. 2, our grasp planning pipeline primarily consists of a pre-trained vision-based planner (GG-CNN) trained on the Cornell Grasp Dataset [23], followed by an RL agent that takes in other inputs from the environment together with the GG-CNN grasp plan, controls the robot, and executes the grasp.

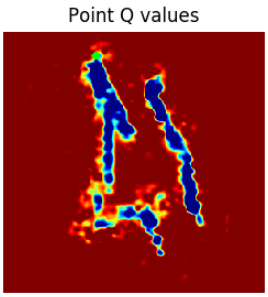
A. Vision-based Grasp Planner

The visual perception part of the system is a combination of the wrist-mounted RGB camera and the GG-CNN based network [2]. The RGB camera produces a 640×480 image, which is then cropped and segmented to a 300×300 size and fed into the network. It is trained on the annotated Cornell Grasping Dataset [23], and generates a pixel-wise grasp vector $\tilde{g}$ for each input image. It is trained for top-down grasps in a table-top setting as shown in Fig. 1, where the effective movement of the robot's wrist is in 2.5D (XY plus height), with the orientation kept vertical as shown in Fig. 1. The original implementation of GG-CNN uses ROS and Gazebo [24], while the RL agent was trained using an OpenAI Gym environment in PyBullet. We therefore re-implemented GG-CNN in PyBullet (Fig. 4). This implementation retained the pre-trained models using the



(a) Raw RGB image

(b) Cropped and segmented image



(c) Pixel-wise grasp quality metric (Q)

Fig. 4: PyBullet implementation of the GG-CNN vision-based grasp planning module: using an RGB image as the input, and providing robot wrist positions and angles as the output using a grasp quality metric.

Cornell Grasping Dataset [23] with the interface for image-based queries made compatible with PyBullet.

For the setup shown in Fig. 1, a single image of the target object is taken at the start of each training episode, and the grasp vector $\tilde{g}$ is computed by the GG-CNN. $\tilde{g}$ consists of the following quantities:

$$\tilde{g} = [\tilde{s}, \tilde{\phi}, \tilde{w}, Q] \tag{5}$$

Here $\tilde{s}$ is the position in XY coordinates of the RGB image, $\tilde{\phi}$ is the rotation angle of the robot's wrist joint in image space, $\tilde{w}$ is the gripper width for a two-fingered parallel jaw gripper, and Q is the grasp quality metric for the particular wrist placement. Q is a measure of the probability of a successful grasp. The corresponding grasp vector in the world coordinate frame, g, can be computed through linear transformations based on the internal camera parameters. In the PyBullet simulation environment used in this work, the camera parameters are considered to be those of an Intel RealSense D435i, with the depth sensing replaced by directly measuring the depth map from the simulation.

$$g = [\vec{p}, \phi, w, Q] \tag{6}$$

Here $\vec{p} = (x, y, z)$ is the Cartesian position of the wrist, ϕ is the rotation angle of the robot's wrist joint (roll angle about the vertical axis), and w and Q are the same as before. Of these quantities, we take $G = [(x, y), \phi, Q]$ as a relevant grasp vector for the soft gripper, since the effective "grip width" is discovered by the RL agent, and the Z-position of the wrist is an independently observed as well as controlled quantity. Additionally, instead of using only the optimal grasp vector as in [2], we take the top five grasp vectors generated by the GG-CNN as part of the observed state for the RL agent, in order to obtain more insights into the shape and features of the target object. It was found during initial trials during development of the model that the single top grasp vector alone would sometimes miss a natural grasp pose, especially for the more adversarially-designed objects.

B. RL Agent

The Reinforcement Learning agent used here is based on the Soft-Actor Critic (SAC) [25] algorithm, with the network weights shared between the policy and the Q-function estimation network. SAC was chosen here as it has been applied with considerable success in robotic tasks [26]. It was implemented using the Stable Baselines [27] package. The policy network architecture consists of two hidden layers of 64 Multi-Layer Perceptrons each, with ReLU activation. A layer normalized version of this policy is used for better performance [28], the entropy regularization coefficient is learned automatically with the initial value of 0.1, and all other learning parameters are kept at their default values.

The observed state vector S is defined as follows:

$$S = \begin{bmatrix} x_w \\ y_w \\ \phi_w \\ \vec{G} \\ \vec{\kappa} \\ \vec{\tau} \end{bmatrix} \tag{7}$$

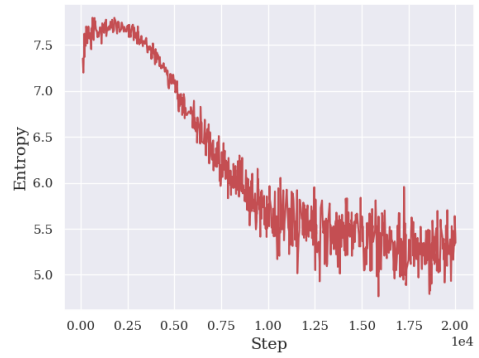
Here (x_w, y_w) are the XY coordinates of the robot's wrist, ϕ_w is the wrist rotation angle, $\vec{G} = [G_1, \ldots, G_5]^T$ are the top five outputs from the GG-CNN arranged in descending order of Q metrics, $\vec{\kappa} = [\kappa_{i,j}]^T$ and $\vec{\tau} = [\tau_{i,j}]^T$ are the collected curvatures and torsions of the fingers, arranged as the j^{th} segment of i^{th} finger.

The RL agent generates the following control action:

$$A = \begin{bmatrix} h & \phi & \vec{F} \end{bmatrix}^T \tag{8}$$

Here h is the height of the robot's wrist above the surface of the table, constrained between 11 cm and 20 cm. The corresponding robot joint angles are computed during execution based on the inverse kinematics (IK) implementation for the TM5-700 in PyBullet. It is constrained to position-only IK as the wrist remains vertical throughout. Only the rotation of the wrist about the vertical axis, ϕ varies, which influences the relative positioning of the fingers with respect to the target object. This control input has a symmetry of $\pi/2$ due to the square design of the gripper, leading to the range of $[0, \pi/2]$ for ϕ. Finally, $\vec{F} = [f_1, f_2, f_3, f_4]^T$ is the vector of control inputs for each of the four fingers. As described earlier, each segment of a finger is given the same joint angle reference, leading to an overall expected quadratic curvature profile.

(a) Episode reward



(b) Entropy

Fig. 5: Training curves for the RL agent: a) Episode reward, b) Entropy loss, over 2×10^4 steps.

The reward function is defined as follows:

$$R = \sum_{i=1}^{4} w_1 e^{-w_2 d_{c,i}} + \delta_h w_3 \left[\frac{\Delta h - h_t}{h_f - h_t}\right]^2 \quad (9)$$

In the first term, $d_{c,i}$ is the distance between the spherical tip of the i^{th} finger and the geometric center of the target object. This term penalizes the tips of the fingers being away from the object, favoring grasps where the the finger tips closely wrap around the object.

The second term rewards lifting the object off the table, a measure of how tightly the object has been grasped. The objective is to raise the object to the final height h_f above the table (taken here to be 20 cm). Δh is the instantaneous difference in height of the target object above its initial Z-position on the table at the start of a training episode. h_t is the threshold height (here 2 cm) above which this reward term is activated:

$$\delta_h = \begin{cases} 1, & \text{if } \Delta h \geq h_t \\ 0, & \text{otherwise} \end{cases}$$

The parameters $[w_1, w_2, w_3]$ are [5, 1, 20] respectively, chosen empirically through results from initial trials while building the model. The RL agent is trained on a subset of 100 objects from the set of random CAD models described in [4] and supplied with the PyBullet random URDF library. These objects were pruned from the overall set to ensure that they are neither too large nor too small to fit in the gripper. This set included a combination of roughly prismatic objects, as well as objects with geometric features that would be adversarial for a parallel-jaw gripper. An object was selected at random in each episode, with each episode lasting for 50 steps, and overall training conducted for 2×10^4 steps. The resulting mean training curves for three seeds are shown in Fig. 5.

V. Results

The SAC-based RL agent trained in the previous section is compared against a baseline grasping strategy that considers a synergy of all four fingers working in tandem.

In the baseline, the robot's wrist is aligned vertically with the center of the target object. The wrist is rotated by and angle ϕ_w from the optimal output of the GG-CNN, and all four fingers are given the same command, i.e. $\vec{F} = [0.2, 0.2, 0.2, 0.2]^T$, for closing. The height of the wrist h, is computed by measuring the distance between the Z-coordinate of the center of the target object and the base of the cubical box on which the fingers are attached. To release the object onto the red target region on the table (Fig. 1), the same gripper opening command $\vec{F} = [-0.2, -0.2, -0.2, -0.2]^T$ is given in both the baseline condition as well as the for the RL agent.

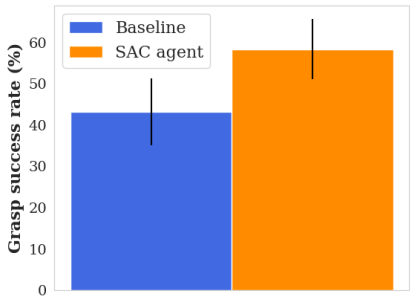


Fig. 6: Comparison of grasp success rates for the SAC-based RL agent and the baseline grasp synergy.

During testing, the RL agent's action A is frozen to the current values of h, ϕ and $\vec{F}$ either when the target object is lifted beyond 5cm or when the maximum number of steps (50 steps) is reached in the episode. The test setup consists of another 100 objects from the PyBullet random URDF library not included during training. A test was considered successful if the target object was picked from its starting position and dropped onto the red target region.

This test protocol was conducted five times for the 100 target objects with both the grasping strategies (baseline synergy and RL agent). The mean success rate for the baseline grasp synergy was 43.2%, while it improved to 58.4% for the SAC-based RL agent (Fig. 6).

VI. Conclusion

In this paper, we describe an attempt to transfer a vision-based CNN approach for grasping developed for two-fingered parallel jaw grippers onto grasping using a soft, deformable gripper using reinforcement learning. We adapted an existing GG-CNN algorithm to PyBullet and incorporated it in an OpenAI Gym environment for training an RL agent to perform top-down grasps of single objects on a table-top.

While the trained RL agent was moderately more successful than a baseline strategy based on pinch grasping, this performance was achieved in a simulation environment where the contact dynamics may not be completely accurate compared to a physical setup. Also, the grasp success rates are not yet sufficient for this system to be deployed in real tasks. Altering the RL architecture, or using model-based or multi-agent RL approaches are being explored to improve the success rate.

The curvature and torsion of the fingers was directly computed from position tracking of the segments in simulation. In a physical implementation, these quantities would need to be estimated from visual inputs, embedded sensors, or high-fidelity nonlinear dynamics models obtained from system identification and incorporating finite element methods. While this strategy was developed for a particular four-fingered gripper design, it can be adapted to any finger configuration by suitably adapting the state and action spaces, and retraining the RL agent.

In ongoing and future work, we aim to conduct a sim-to-real transfer of an RL agent onto a physical setup with a soft robotic gripper, testing initially for open-loop top-down single object grasps, and moving forward to in-hand manipulation, grasping in clutter, and fully 3D grasp planning during mobile manipulation with closed-loop visual feedback.

Acknowledgment

The authors would like to thank the Robotics and Autonomous Systems group at TCS R&I for their valuable suggestions regarding the methods applied in this paper.

References

[1] D. Prattichizzo and J. C. Trinkle, "Grasping," in *Springer Handbook of Robotics*. Springer, 2016, pp. 955–988.

[2] D. Morrison, P. Corke, and J. Leitner, "Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach," *arXiv preprint arXiv:1804.05172*, 2018.

[3] E. Coumans and Y. Bai, "PyBullet, a Python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.

[4] D. Quillen, E. Jang, O. Nachum, C. Finn, J. Ibarz, and S. Levine, "Deep reinforcement learning for vision-based robotic grasping: A simulated comparative evaluation of off-policy methods," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6284–6291.

[5] X. Chen, X. Zhang, Y. Huang, L. Cao, and J. Liu, "A review of soft manipulator research, applications, and opportunities," *Journal of Field Robotics*, 2021.

[6] M. G. Catalano, G. Grioli, E. Farnioli, A. Serio, C. Piazza, and A. Bicchi, "Adaptive synergies for the design and control of the pisa/iit softhand," *The International Journal of Robotics Research*, vol. 33, no. 5, pp. 768–782, 2014.

[7] R. Deimel and O. Brock, "A novel type of compliant and underactuated robotic hand for dexterous grasping," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 161–185, 2016.

[8] B. S. Homberg, R. K. Katzschmann, M. R. Dogar, and D. Rus, "Robust proprioceptive grasping with a soft robot hand," *Autonomous Robots*, vol. 43, no. 3, pp. 681–696, 2019.

[9] S. Abondance, C. B. Teeple, and R. J. Wood, "A dexterous soft robotic hand for delicate in-hand manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5502–5509, 2020.

[10] A. Gupta, C. Eppner, S. Levine, and P. Abbeel, "Learning dexterous manipulation for a soft robotic hand from human demonstrations," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 3786–3793.

[11] C. Choi, W. Schwarting, J. DelPreto, and D. Rus, "Learning object grasping for soft robot hands," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2370–2377, 2018.

[12] S. Joshi, S. Kumra, and F. Sahin, "Robotic grasping using deep reinforcement learning," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2020, pp. 1461–1466.

[13] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[14] S. Levine and V. Koltun, "Guided policy search," in *International conference on machine learning*. PMLR, 2013, pp. 1–9.

[15] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics," *Advances in neural information processing systems*, vol. 27, 2014.

[16] M. Pozzi, S. Marullo, G. Salvietti, J. Bimbo, M. Malvezzi, and D. Prattichizzo, "Hand closure model for planning top grasps with soft robotic hands," *The International Journal of Robotics Research*, vol. 39, no. 14, pp. 1706–1723, 2020.

[17] T.-I. Chen, "NSC Short Project on Robotic Grasping with Reinforcement Learning," https://github.com/Tung-I/RoboticGrasper, 2020.

[18] M. A. Graule, C. B. Teeple, T. P. McCarthy, G. R. Kim, R. C. S. Louis, and R. J. Wood, "Somo: Fast and accurate simulations of continuum robots in complex environments," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 3934–3941.

[19] M. Graule, "SoMo," https://github.com/GrauleM/somo, 2021.

[20] "Soft Robotics Inc. - mGrip modular gripping system," https://www.softroboticsinc.com/.

[21] D. Bruder, C. D. Remy, and R. Vasudevan, "Nonlinear system identification of soft robot dynamics using koopman operator theory," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6244–6250.

[22] A. Koivikko, E. Sadeghian Raei, M. Mosallaei, M. Mantysalo, and V. Sariola, "Screen-printed curvature sensors for soft robots," *IEEE Sensors Journal*, vol. 18, no. 1, pp. 223–230, 2018.

[23] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.

[24] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[25] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[26] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: lessons we have learned," *The International Journal of Robotics Research*, vol. 40, no. 4-5, pp. 698–721, 2021. [Online]. Available: https://doi.org/10.1177/0278364920987859

[27] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable baselines," https://github.com/hill-a/stable-baselines, 2018.

[28] J. Xu, X. Sun, Z. Zhang, G. Zhao, and J. Lin, "Understanding and improving layer normalization," *Advances in Neural Information Processing Systems*, vol. 32, pp. 4381–4391, 2019.