Online Deep Reinforcement Learning on Assigned Weight Spaghetti Grasping in One Time using Soft Actor-Critic

Prem Gamolped[1], Sakmongkon Chumkamon[1], Chanapol Piyavichyanon[2], and Eiji Hayashi[1]
[1]Department of Mechanical Information Science and Technology, Kyushu Institute of Technology,
680-4 Kawazu, Iizuka, Fukuoka 820-8502, Japan
[2]Department of Creative Informatics, Kyushu Institute of Technology,
680-4 Kawazu, Iizuka, Fukuoka 820-8502, Japan

Abbe Mowshowitz[3]
[3]Department of Computer Science, The City College of New York,
160 Convent Avenue, New York, NY 10031, USA

E-mail: gamolped.prem129@mail.kyutech.jp, m-san@mmcs.mse.kyutech.ac.jp,
piyavichayanon.chanapol402@mail.kyutech.jp, haya@mse.kyutech.ac.jp, amowshowitz@ccny.cuny.edu
http://www.kyutech.ac.jp/

Abstract

Artificial Intelligence and Robotics have become essential and widely used to package food. Packaging an assigned weight spaghetti into a lunch box at one time can be difficult. This paper proposes a solution for one-time grasping using Deep Reinforcement Learning (DRL) based on the Soft Actor-Critic algorithm on the manipulator. Spaghetti detection and segmentation are implemented from the RGB-D camera for the observation. We conclude that the experiment shows the effectively grasped result can almost succeed within 10% of the target weight in the experimental environment.

Keywords: One-Time Grasping, Deep Reinforcement Learning (DRL), Soft Actor-Critic, RGB-D Camera

1. Introduction

In the past years, the technology of robotics in the food industry has tremendous advanced for the task mainly focused on packing, picking, palletizing the food, etc. Many researchers implement artificial intelligence technology with robotics to create self-decision based on observation. The task is repetitive since in packaging the specific amount of spaghetti into the lunch box, the human can pick up the spaghetti and measure the weight before put into the lunch box, as shown in Fig 1. This work proposes a solution for one-time spaghetti grasping using a self-learning system called Reinforcement Learning (RL) based on the off-policy learning RL called

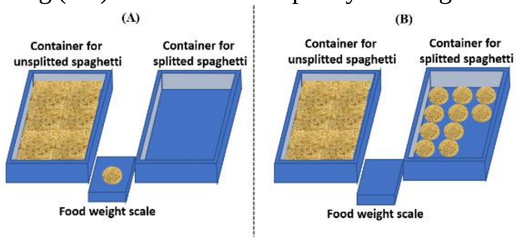
(A)
(B)
Container for unsplitted spaghetti
Container for splitted spaghetti
Container for unsplitted spaghetti
Container for splitted spaghetti
Food weight scale
Food weight scale

Fig. 1. On the A-side, the worker picks spaghetti from the left container, measures weight, and puts it in the right container, as shown on the B-side.

Soft Actor-Critic (SAC). The spaghetti detection and segmentation system from the RGB-D camera are implemented to recognize the state for observation of RL. We introduce the learning and evaluation system framework for a one-time grasping of the following task. The result of our proposed solution shows that the grasped result can almost succeed within 10% of the target weight.

2. Related Work

The spaghetti grasping task in a manipulated robot can be enhanced by deciding the action to take based on the robot's environment, also called self-decision capability. The RL learning system is proposed in many research papers because it can perform the learning task from the ground up without giving any data to learn at the initial state.

2.1. *System Configuration*

Performing the spaghetti grasping task on the manipulated robot. The controlling, perceiving, and feedback system should be considered, especially the grasping system in the part of the gripper that should match with an object to grasp. Spaghetti is regarded as a soft-body object that can be damaged quickly and causes the worse result in training. The robot arm we use is the Motoman SIA5F articulated arm, as shown in the Fig. 2.

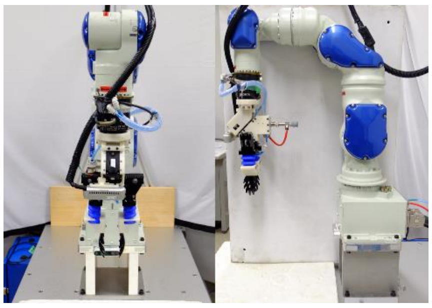

Fig. 2. The Motoman SIA5F, 7-Axis Articulated arm robot with 5KG Payload. Attached with gripper system for the spaghetti grasping task.

Fig. 3 shows the gripper system that consists of the following

- The Twintool gripper can switch between two attached tools motorized using Servo motor
- Soft Robotics gripper is the pneumatic actuated soft gripper for food grasping and an additional noodle tong for spaghetti grasping.
- The Vacuum tool to vacuum the flat face food such as ham, cut bread, bologna, etc.
- The RGB-D camera will provide the color and depth image for the robot's perception.

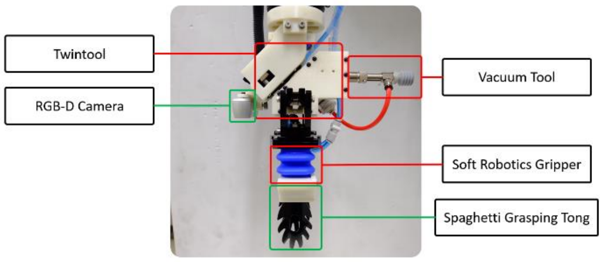

Fig. 3. Gripper system

We created the spaghetti detection system to identify the probability of being the spaghetti and the spaghetti instance segmentation to get a centroid of the shape to help perform the limit of the environment in RL for the agent. We collected the lunch box and spaghetti dataset then manually labeled and annotated it using the COCO annotator. The dataset was trained using the Hybrid Task Cascade (HTC) method on the MMDetection framework. Fig. 4 shows the vision system for spaghetti. The left is the target box, and the right side is the spaghetti container.

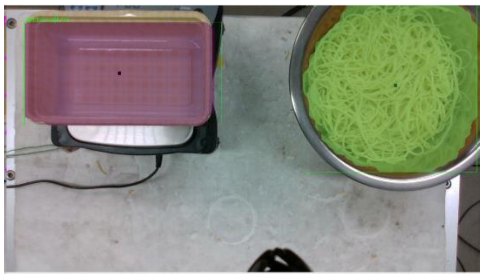

Fig. 4. Spaghetti detection and instance segmentation system

2.2. *Deep Reinforcement Learning*

The self-learning system that does not require any instruction to take action; instead, the agent will perceive the environment from the sensor and take action based on the observation to get the maximum reward. The agent will do the trial-and-error, which yields the positive or negative reward based on the action that agent takes through the reward function. DRL mainly consists of two parts shown in Fig. 5

- Agent (Human or Robot)
- Environment

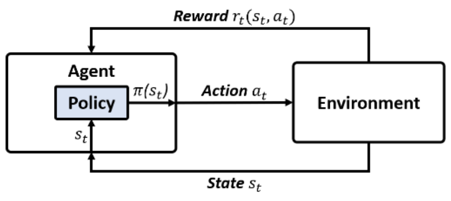

Fig. 5. Spaghetti detection and instance segmentation system

In DRL, we will consider in timestep t according to the finite-horizon discounted Markov Decision Process (MDP) wherein each t; the agent is going to get state s_t from the environment, the policy function is going to map from what the agent observes to action a_t, in this timestep t, the reward $r_t(s_t, a_t)$ will determine the good or bad that the agent took action. In our research, timestep t is set to be 1. Thus episode length will be one. The method of DRL we use is Soft Actor-Critic (SAC) [1]. The SAC is the off-policy algorithm that can handle the complex environment. The algorithm is well-known in the features of the following:

- SAC has the entropy regularization that agent will get the best of the trade-off between explore-exploit, which will encourage the agent to explore more.
- A consequence of the explore-exploit trade-off yields the training faster later because the agent gets to know a lot in the environment.
- SAC can work on the continuous action space.

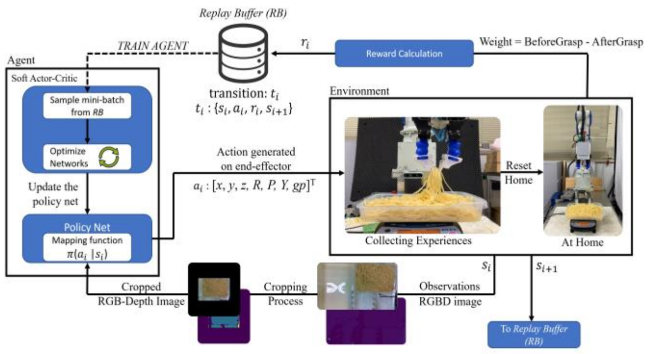

Fig. 6. The DRL framework of the training and evaluation system on one-time spaghetti grasping task using DRL

The proposed training framework of one-time spaghetti grasping will imitate the MDP process that wherein each timestep t; we will take an RGB-D image of the spaghetti and do the image processing by cropping unwanted area to get the image that has a high percent of spaghetti included as a state t, and the policy network we use is the convolutional neural network (CNN) to map from state t to action a_t, the action space of our framework consists of the following values: position, orientation, and gripper width of the end-effector; thus, our action a_t is x, y, z, R, P, Y, gp. The command will be sent to the sequence of the motion we create for picking the spaghetti and measuring the weight, then randomly put spaghetti back to the container. The reason to randomly put is to avoid the same observation every timestep t. The reward $r_t(s_t, a_t)$ is calculated from the following formula.

$$r_t(s_t, a_t) = \frac{(target - |measured - target|)}{target} \qquad (1)$$

The agent will calculate the reward function based on Eq. (1) if the measured or the weight from grasped spaghetti is in 10% of the target weight. In our research, the target weight is 50-gram; thus, the reward function will be activated when the measurement is between 45-55 grams. Otherwise, the $r_t = -0.5$ and after the reward, the next state s_{t+1} will be taken with the cropping process to complete the transition for replay buffer to complete one timestep.

3. Experiment

To perform the experiment of the one-time spaghetti grasping, we set up the experiment inside the laboratory environment.

3.1. *Setup the Experiment*

We will start to prepare spaghetti for 300-gram and boil for 7 minutes; after that, we will shake off the water until almost dry and put the oil in a bit to prevent sticking spaghetti because it can damage and make training results worse. Fig. 7 shows the prepared environment and spaghetti after boiling.

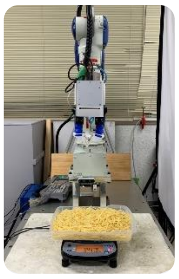

Fig. 7. Prepared environment before starting training

3.2. *Training*

We have set the training for two sessions; each session run through 2,000 episodes, the first one of A, we use the reward function that has no negative and second run of B, the negative reward is included as shown in Eq. (1). However, the spaghetti will get dried very fast. Thus, the spaghetti entangling problem can happen, and the training can worsen because the physical weight will change a lot from its undamaged spaghetti. To solve this problem, we spray a little bit of water onto the spaghetti face every 20 minutes and change new spaghetti every 200 episodes to make sure of training the undamaged spaghetti. Fig. 8 shows the training result of the two 8sessions described above. After that, we get the Episode Reward Mean to conduct the training trend. We can see that the model that negative reward is included is better because we can filter out the unimportance of action that led to always getting the positive reward; also, we can say that it is a Sparse Reward. Table 1 shows the comparison result of two training sessions. Table 2 shows the Mean Absolute Percent Error (MAPE) which is used to measure the percent error of model prediction.

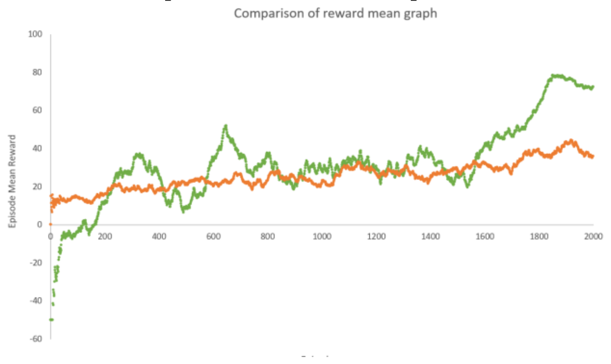

Fig. 8. The reward mean graph from two training sessions

Table 1. Training result comparison of two sessions

Parameters	A [g]	B [g]
Mean	46.72	48.90
Absolute Mean	19.29	14.08
Standard Deviation	23.41	17.91

Table 2 shows that the model with the negative reward included could do lower the MAPE, which mean that with the lower value, it is possible to get near the 10% of target weight

Table 2. MAPE comparison of two sessions

Parameters	A [%]	B [%]
MAPE	38.97	28.44

4. Evaluation

The same framework is used to evaluate the model with lower MAPE or model B to show how accurate the model is; thus, we set up the same environment and proceed with the same training step. We acquired only 100 episodes of the session, calculated how many samples are in the 10% of the target weight, and plotted the evaluated model's distribution. Fig. 9 shows the distribution of the evaluated model. Table 3 shows the following described above.

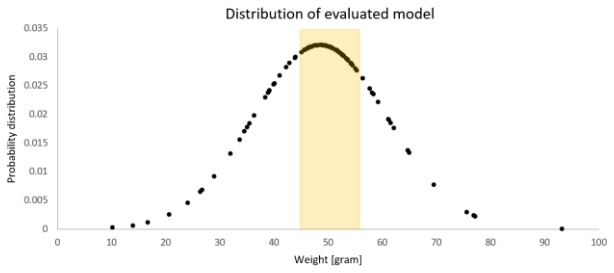

Fig. 9. The distribution of evaluated model

Table 2. Result of evaluated model

Parameters	B	unit
Mean	48.52	grams
Absolute Mean	8.31	grams
Standard Deviation	12.41	grams
Grasp in the range of acceptable error	58	times

5. Conclusion

In this research, we proposed the solution to grasp a one-time spaghetti at the assigned, and the result shows that we can almost succeed within 10% of the target weight. However, there is still a problem that spaghetti gets to dried quickly and it can cause the entangling problem and can worsen the training result. Also the reward function should have the negative reward to prevent the sparse reward. In the future work, we will try to control the condition of spaghetti such as time and the humidity of spaghetti.

References

1. [1801.01290] Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor (arxiv.org)

Authors Introduction

Prem Gamolped

He received bachelor degree in Engineering in 2021 from Robotics and Automation Engineering, King Mongkut's University of Technolgoy North Bangkok in Thailand. He is currently a Master student at Kyushu Institute of Technology and conducts research at Hayashi Laboratory.

Dr. Sakmongkon Chumkamon

Dr. Sakmongkon Chumkamon received Doctor of Engineering degree from Kyushu Institute of Technology in 2017. He was a postdoctoral researcher at Guangdong University of Technology in 2017-2019. Presently he is a postdoctoral researcher in Kyushu Institute of Technology since 2019. His research interests include factory automation robots and social robots.

Mr. Chanapol Piyavichayanon

He received bachelor degree in Engineering in 2021 from Mechanical Engineering, Chulalongkorn University in Japan. He is currently a Master student at Kyushu Institute of Technology and conducts research at Koga Laboratory.

Prof. Abbe Mowshowitz

Prof. Abbe Mowshowitz received the Ph.D. degree from University of Michigan in 1967. He has been professor of computer science at the City College of New York and member of the doctoral faculty at the Graduate Center of the City University of New York since 1984. His current research interests lie in two areas are organizational and managerial issues in computing, and network science. In addition to teaching and research, He has acted as consultant on the uses and impacts of information technology (especially computer networks) to a wide range of public and private organizations in North America and Europe.

Prof. Eiji Hayashi

Prof. Eiji Hayashi is a professor in the Department of Intelligent and Control Systems at Kyushu Institute of Technology. He received the Ph.D. (Dr. Eng.) degree from Waseda University in 1996. His research interests include Intelligent mechanics, Mechanical systems and Perceptual information processing. He is a member of The Institute of Electrical and Electronics Engineers (IEEE) and The Japan Society of Mechanical Engineers (JSME).