Master's Thesis of Science in Engineering

Task-Oriented Design Through Deep Reinforcement Learning

심층 강화학습을 이용한 과제 기반의 디자인 방법론

February 2019

Graduate School of
Convergence Science and Technology
Seoul National University
Department of Transdisciplinary Studies

Junyoung Choi

Task-Oriented Design Through Deep Reinforcement Learning

Nojun Kwak

Submitting a master's thesis of Public Administration

December 2018

Graduate School of
Convergence Science and Technology
Seoul National University
Department of Transdisciplinary Studies

Junyoung Choi

Confirming the master's thesis written by
Junyoung Choi

February 2019

Chair	_____	(Seal)
Vice Chair	_____	(Seal)
Examiner	_____	(Seal)

Abstract

We propose a new low-cost machine-learning-based methodology which assists designers in reducing the gap between the problem and the solution in the design process. Our work applies reinforcement learning (RL) to find the optimal task-oriented design solution through the construction of the design action for each task. For this task-oriented design, the 3D design process in product design is assigned to an action space in Deep RL, and a desired 3D model is obtained by training each design action according to the task. By showing that this method achieves satisfactory design even when applied to a task pursuing multiple goals, we suggest the direction of how machine learning can contribute in design process. Also, we have validated with product designers that this methodology can assist the creative part in the process of design.

Keyword : Task-oriented Design, Design Through Machine Learning, Reinforcement Learning, Deep Learning

Student Number : 2017-22087

Contents

Chapter 1

Introduction

This paper suggests a methodology of helping designers to obtain morphological insights with the use of deep reinforcement learning. Within the constructive design process, design methodologies have evolved to find the points of human inconvenience or the problems contained in the use of the product [1]. In the problem solving process, a design solution for the defined problem is searched for that can be applied to the actual design. Many designers have suggested various methods to solve these design problems. [2, 3]. However, since these methods usually require high cost in terms of time and space, many researchers have difficulties in finding a good solution.

Meanwhile, the role of computer science in the design process has been changed. Computer scientists collaborate not only in the development of design tools that designers can use directly for designing products, but also in various forms that help designers with computer simulation or machine learning algorithms before the mock-up stage [4, 5, 6, 7, 8]. However, these approaches only use computer science as a rule-based assistant, not a creative designing tool. Dreamsketch [9], on the other hand, suggested the

methodology to obtain multiple 3D design solutions based on the context created from the sketch phase. But still, because it is applied only after all the analysis has been done, rather than used for understanding the context of design, it is of little help in the design process and does not directly address the underlying problem solving. Pahn et al. [10] is another good example that attempted to assist the design process using machine learning, but it is very costly because it requires the process of dividing the product and the corresponding design problem into small objects and assigns a role to each object.

Different from the above mentioned methods, we have chosen algorithms that can more directly understand the design process in order to suggest a more direct and lower-cost methodology to designers at the problem-solution bridge stage. The reinforcement learning algorithm, which is becoming a hot topic in computer science, is an algorithm that learns the best action (to get the best reward) in a provided environment. Even if we do not give detailed information about the intermediate process leading to the best reward, we can learn to get the best action possible. Through this, we devised a framework that can design based on a given task.

In this paper, we apply the reinforcement learning algorithm to product design and present a link that enables the computer to directly find the task-oriented solution through the problem. The whole proposed methodology is shown in Figure 1. When problems from design research is given, we define tasks, 3D simulation environments and reinforcement learning environments. Tasks and 3D simulation environments are processed in BLENDER [11] and linked to the reinforcement learning algorithm. In the figure, this
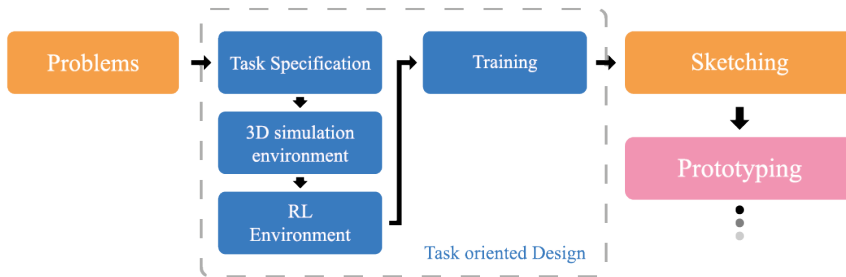
2

Figure 1: The proposed method of task-oriented design through deep reinforcement learning in the product design precess. With task specification, 3D simulation and a reinforcement learning environment, our methodology propose a methodology that directly assists creative part of the product design process.

paper deals with the processes denoted in the dashed-line box.

Zhu et al. [8] is also a good example of a deep learning algorithm for product interpretation. However, we go one step further and present a methodology that allows the computer to design itself based on the understanding of the product or task. More specifically, we have chosen to design a pot with a couple of design objectives. By tackling this problem, we will discuss how the reinforcement learning algorithm finds solutions in order to achieve high scores in a given task. To enable this, we define the design process as an action space that can be understood by the computer. We will also cover how we can finally use the generated output by giving morphological intuition to product designers.

Chapter 2

Related Works

2.1 Constructive Design

In various study of product design process [1, 12], the authors discuss about the process of constructive design research which is initiated by formulating a research question out of an existing theory or philosophy, then investigate the question through a process of making and designing artifacts.

For constructive design, user research should be proceeded first. With studying users and products, designers get several insights that should be applied to their final design. After studio work, constructive design researchers develop designs, which begins with sketchy ideas and mock-ups. In this stage, usually hundreds of mock-ups are made by designers, which costs a lot of time and efforts.

2.2 Reinforcement Learning

Reinforcement learning is an algorithm that learns which actions to take to maximize rewards in a given environment. In reinforcement learning, we define and solve problems with the framework of Markov Decision Pro-

cesses (MDPs). MDPs consist of the environment and the agent. They interact each other at every continual time index $t = 0, 1, 2, \cdots$, and the agent tries to achieve a given goal.

Specifically, the agent receives information about the state from the environment and takes action to obtain maximum rewards in the current state, and the action is determined by the policy. The policy is defined as a probability distribution $(\pi(a|s))$ that is of available action outputs in a given state [13]. The environment outputs rewards as a result of agent's action and this process repeats. In the end, trajectories such as $s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, \cdots$ can be obtained by the interaction of the agent and the environment. State changes are determined by stationary transition dynamics distribution $p(s_{t+1}|s_t, a_t) = p(s_{t+1}|s_0, a_0, \cdots, s_t, a_t)$ which follows markov properties [14].

Deep Q-Network(DQN) has achieved surprising performance in the Atari2600 task learning environment, where the action-value function ($q(s_t, a_t)$; Q-function) was approximated by deep neural network [15]. The Q-function estimates the maximum achievable cumulative rewards for a current state s_t with action a_t. In DQN, the output of the Q-function is an one-hot vector form of the discrete action space, which prohibits DQN from being applied to continuous action space environments.

Deep deterministic policy gradient (DDPG) [16] used the Actor-Critic method [17] to overcome the disadvantages of DQN in the continuous action space. Trust region policy optimization (TRPO) [18] then used the Kullback-Leibler divergence [19] as a constraint to resolve the unstable result of policy update due from the fixed step size in DDPG. In addition, proximal policy optimization algorithm (PPO) [20] proposes a clipped sur-

rogate objective function to alleviate the complex computational require-
ment in TRPO. In this experiment, we used PPO as a main algorithm to
solve the task-oriented optimization problem.

Chapter 3

Environment

3.1 Task Specification

In this study, to verify the effectiveness of RL in the design process, we selected the *cylindrical pot* shown in Fig. 2 as a basic design for its simple but flexible form. Starting from a cylinder-shaped pot, the agent tries to maximize the cumulative reward for a specific task by taking actions which is defined as increasing or decreasing the diameter of the pot at different heights. The first task we consider is pouring as much water as possible in the pot into another cup according to the primary purpose of the pot. In addition, we assumed a shaking situation as a second task. Here, the agent tries to keep water from the shaking pot. We named each environment as 'pouring environment' and 'shaking environment', respectively. After training each task successfully, we also show that simultaneously training both tasks is possible despite these two tasks have conflicting features.
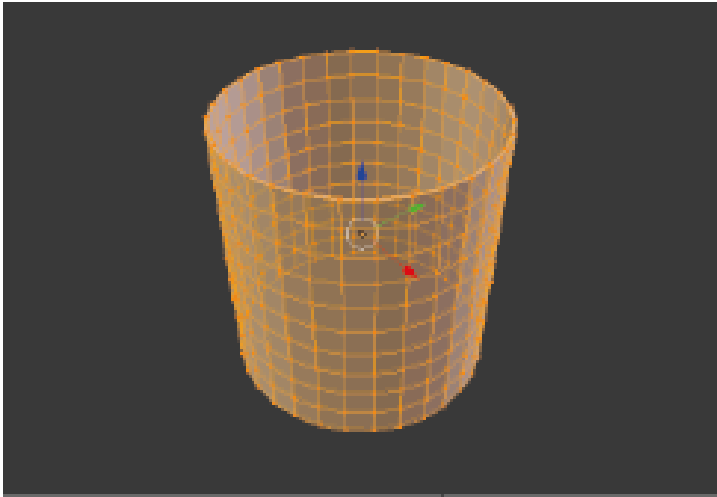
Figure 2: Initial state of pot design drawn in the BLENDER environment.

3.2 3D Simulation Environment

The existence of simulation environment has been one of the main reasons for the success of deep reinforcement learning. Mnih et al. [15] was able to find the optimal policy by training in the ATARI2600 game environment through Arcade Learning Environment [21]. The role of the simulation is also important in our task. Without simulation, we have to repeat the inefficient process of designing the pot each time, outputting the product, and experimenting.

We used an open-source 3D modeling tool, BLENDER, to construct a reinforcement learning environment. The reason for using the BLENDER is that it allows fluid simulation through embedded *particle systems* and can control and output all the available information on the environment via
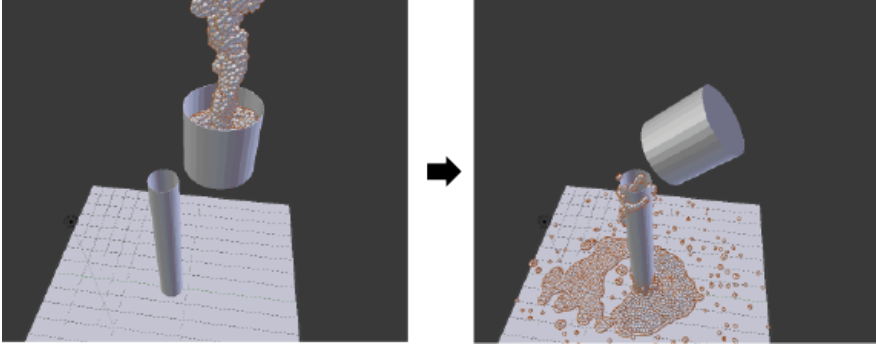
Figure 3: Initial step in the pouring environment (left). We simulate the pouring operation by tilting the pot and filling the cup with water (right). The amount of water in the cup is rewarded to the agent which further tries a designing action of increasing or decreasing each diameter of the 11 control points. Note that since the pot and the cup are located with some distance, most of the water spill during pouring operation.

python scripts. The initial model of the pot implemented with the BLENDER

is shown in Figure 2.

3.3   Reinforcement Learning Environment

As described in the above section, to apply each modeling design to reinforcement learning, we need to define the state and action space. To define the state and action, we assigned 11 control points along the z-axis of the pot by dividing z-axis into 10 regions. And each cross sectional circle corresponding to each control point consists of 32 points with equal distances. Now the pot consists of 352 ($= 32 \times 11$) points as we can see in Figure 2. The action space is defined as an 11-dimensional vector that controls the radii of 11 circle groups and the state is a vector of 1,056 ($= 32 \times 11 \times 3$) dimension which corresponds to x, y, z coordinates of all points in the pot
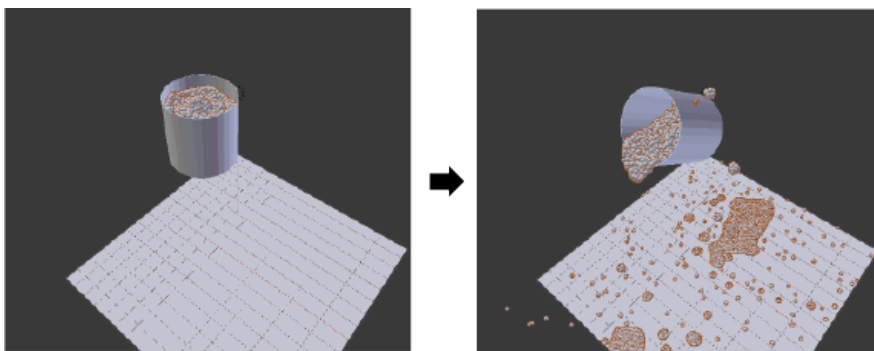
Figure 4: Initial step in the shaking environment (left). We simulate the shaking operation by abruptly tilting the pot from 70° to -70°. The remaining water in the pot acts as a reward in this environment.

design.

As we see in Figure 3, in the pouring environment, the environment consists of a pot containing water and a cup to receive water. At first, the agent takes an action to change the design of the pot from the initial state. The environment simulates a step process and then measures a reward. The step proceeds as follows. With a certain amount of water in the pot, tilt the pot from zero to 130 degrees in the direction of x-axis for 2 seconds. At this time, the amount of water in the cup is measured as a reward.

In the shaking environment shown in Figure 4, the environment only consists of a pot containing water unlike the pouring environment. Most of the step processes are similar to the pouring environment except for the simulation step process. With a certain amount of water in the pot, the environment shakes the pot for thirteen seconds from -70 to 70 degrees in the direction of x-axis. At this time, the amount of water remaining in the pot is measured as a reward.
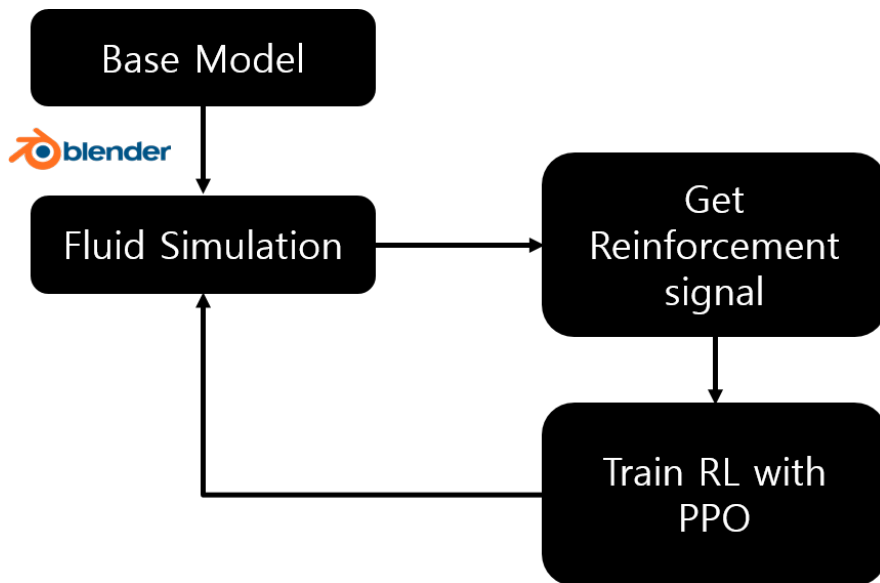
Figure 5: The whole cycle of experiment. 3D model designed by blender directly link to the PPO algorithm in python so that each simulation result can be calculated and use while learning.

Chapter 4

Experiment

In this section, we will show that task-oriented design is possible through experiments not only in the tasks mentioned in the previous section, but also in a multi-task environment where the goal is a hybrid of the two tasks. And then, we will analyze how the computer understands a task and designs a model by examining i) the rewards during learning quantitatively and ii) the result of the pot design qualitatively.

4.1   Pouring Environment

In the pouring environment, the pot performs a task of pouring water into a narrow cup located a certain distance away as shown in Figure 3. If the amount of water in the cup is n_{cup}, and the total amount of water is n_p, the reward is defined as

$$reward_{pour} = n_{cup}/n_p. \tag{4.1}$$

Through this, we designed an experiment to get the maximum amount of water to the cup when tilting the pot. We used PPO as the reinforcement
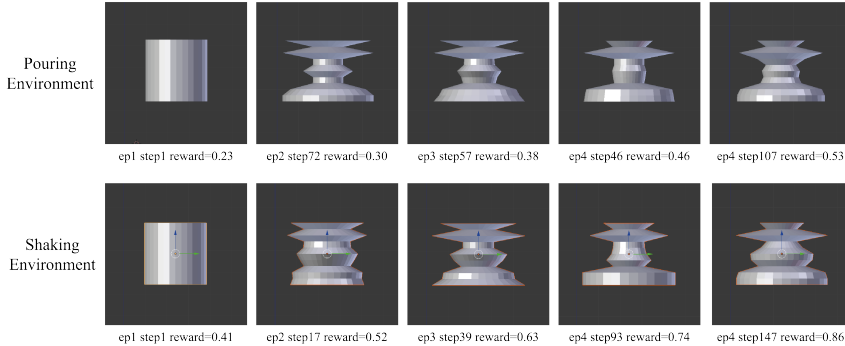
Figure 6: Changes in the model during learning. The first row shows the learning process in the pouring environment, and the second row shows the learning process in shaking environment. The final image of each raw is when the learning has reached its best performance.

learning algorithm. In general, millions to billions of steps are needed for a reinforcement learning model to converge. In our experiment, however, we only used 1,000 steps for training due to the computational bottleneck of the BLENDER simulation. We separated the 1,000 steps into 5 episodes and initialized the pot design every 200 steps to prevent sub-optimality that might occur in reinforcement learning. Through this, we encouraged the agent to make optimal modeling.

4.1.1 Quantitative Analysis

Figure 7 shows the overall reward rise during the training. At the initial state, the reward is quite low due to the distance gap in the *x*-direction between the tip of the pot and the center of the cup. As the training proceeds, the reward increases. The valley for each episode in Figure 7 (at 1, 201, 401, 601, and 801 steps) shows that it starts again from the initial state so that it

13
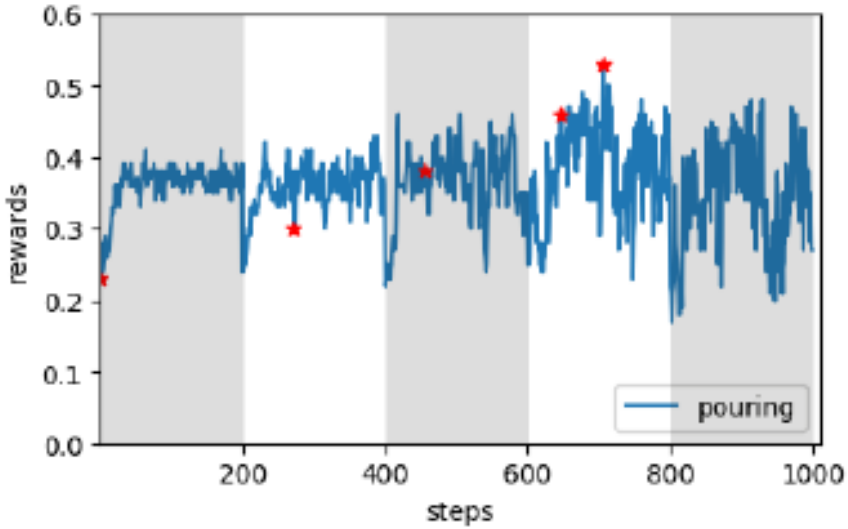
Figure 7: Reward Graph in Pouring Environment. Each shadow on the graph means the full episode of this experiments. In every start point of the episode(200, 400, · · ·), pot design is initialized so that the network can get exploration effect with trained gradient. Each red points on the graph represent steps that shown in the first row of Figure 6.

deviates from the sub-optimality and shows a slight improvement in reward as the learning progresses. Compared to 23% of water in the cup in the initial state, we can see the improvement in performance by containing 53% of water at the end of the training. The final image in the pouring environment of Figure 6 is what the pot design would look like when it got a reward 0.53 at episode 4 step 107.

4.1.2 Qualitative Analysis

If you look at the models created by the deep reinforcement learning algorithm, you can see which tasks the agent want to perform in each step.

14

Table 1: Rewards Table in Tilting Environment. The final reward in the table represent best score in learning. In this case, the best reward was 0.53 in episode 4 step 107.

Episode	Steps	Rewards
1	1	0.23
2	72	0.30
3	57	0.38
4	46	0.46
4	107	0.53

In Figure 6, the first row shows how the model trained from the pouring environment changes. In the initial state, it inevitably fails to aim correctly since the distance from the pot to the cup is far as can be seen in Figure 3. Our agent solve this problem by shaping pot design such that the center is narrow and the head and the bottom are wide, which controls the acceleration of the fluid. After this, the algorithm passes through exploration steps to maximize the reward Eq (4.1). Also, we can see that the head area is resized to maximize the reward.

4.2 Shaking Environment

In the shaking environment, the pot is designed to shed as little water as possible in the environment of shaking the pot. If the initial amount of water in the pot is n_p, and the final amount of water after shaking is n_{pot}, the reward is defined as

$$reward_{shake} = n_{pot}/n_p. \tag{4.2}$$

Through this, we trained the pot to conserve the maximum amount of water in the pot. We used the PPO algorithm like the pouring environment and trained the model for 5 episodes of 200 steps each.

Table 2: Rewards Table in Shaking Environment. The final reward in the table represent best score in learning. In this case, the best reward was 0.86 in episode 4 step 147.

Episode	Steps	Rewards
1	1	0.41
2	17	0.52
3	39	0.63
4	93	0.74
4	147	0.86

4.2.1   Quantitative Analysis

Figure 8 shows the trend of reward as the learning progresses in the shaking environment. The reward tends to decrease slightly after reaching a saturation level in the first three episodes, which indicates that the agent has stuck in the local minimum during training. However, by solving the local minimum problem after episode 4 through exploration, the maximal reward increases. As a result, compared to the initial state which saves only 41% of the water, at the end of the training, the resultant pot is able to keep 86% of the water. The bottom right image of the shaking environment in Figure 6 is when the algorithm gets a reward of 0.86, as recorded in episode 4 step 147.
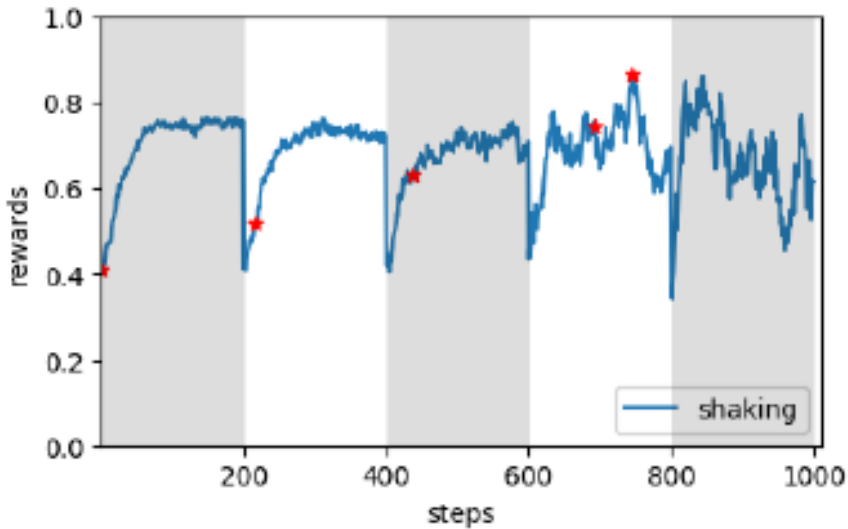
Figure 8: Reward Graph in Shaking Environment. Each shadow on the graph means the full episode of this experiments. Each red points on the graph represent steps that shown in the second row of Figure 6.

4.2.2 Qualitative Analysis

Looking at the changes in the models generated by the reinforcement learning, you can see how the network is trained to protect water. As you can see from the bottom row of Figure 6, the bottom part of the pot becomes larger and larger to keep as much water as possible, and the structure is good for storing water. As training proceeds, it was difficult to store water in the lower part, and the training progressed with a double tube structure. Commonly, there is a barrier structure in the upper part to prevent the pot from splashing by water shaking. Consequentially, we were able to confirm that when the simulation was carried out, it was trained to keep the bouncing water as much as possible in the pot from the large swing of ± 70 degrees.

17

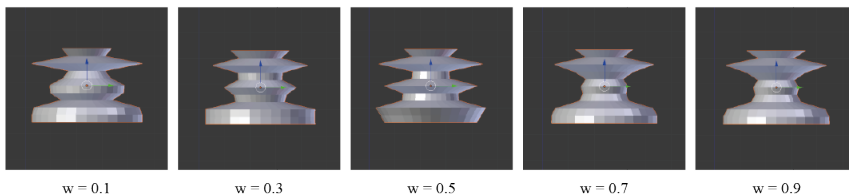|				
w = 0.1	w = 0.3	w = 0.5	w = 0.7	w = 0.9

Figure 9: Final models in cross-learning environment. Each images represent the best performance in five weight parameter with 0.2 difference from 0.1 to 0.9. When the weight parameter is close to 0, the learning is focused on solving shaking environment. On the contrary, the higher weight parameter means that the result of learning will be similar to the solution of the pouring environment.

4.3   Hybrid-Learning

In hybrid-learning, we examined the possibility of pot design that can perform both contradictory tasks. To do this, we defined a new reward which is a weighted sum of the reward in the pouring environment and the reward in the shaking environment as follows:

$$reward_{hybrid} = w \cdot reward_{pour} + (1-w) \cdot reward_{shake} \qquad (4.3)$$

In this equation, w is a weight parameter between 0 and 1. We experimented how the algorithm interprets each task according to five w values in $\{0.1, 0.3, 0.5, 0.7, 0.9\}$.

Table 3: Maximum Rewards in Hybrid-Learning. The hybrid reward is calculated as shown in Eq.4.3. Each rewards are best rewards in chosen weight.

Weight	Episode	Step	Pour	Shake	Hybrid
0.1	4	146	0.32	0.87	0.82
0.3	5	69	0.43	0.83	0.71
0.5	5	51	0.48	0.80	0.64
0.7	4	107	0.55	0.71	0.60
0.9	4	107	0.53	0.71	0.55

4.3.1 Quantitative Analysis

Table 3 indicates how much reward is obtained for different weight parameters. Each episode and step indicates the time when the maximum hybrid reward was achieved for the corresponding weight, and the three rewards (pouring, shaking, and hybrid) are the corresponding rewards at the time. When w is 0.1, according to Eq (4.3), we can see that the shaking environment has more weight on training. As w increases, training is more focused on the pouring environment. As can be seen in Table 3, in the pouring environment, a pouring reward of 0.32 was achieved at the point where the hybrid reward was largest when w was 0.1. As w increased to 0.9, the pouring reward increased to 0.53, which is the best score of the single pouring environment, because the algorithm gave more weight to the pouring environment. Conversely, in the shaking environment, the shaking reward was 0.87 when w was 0.1 and the reward decreased to 0.71 when w was 0.9. In this way, we showed the deep RL algorithm combining the two opposite tasks can train a model that satisfies both tasks.

4.3.2   Qualitative Analysis

Figure 9 shows how the model appears based on the change in the weight parameter w. When the value of w is 0.1, there is a water trap structure at lower position, a narrow entrance, and a barrier structure below the entrance like the model designed in the pure shaking environment. This shows that the training is focused on the shaking environment and trained to maximize water in the pot. On the other hand, when w is 0.9, we can see that the model has trained to create a smooth line in the middle like the model designed in the pouring environment and flows the water as easily as possible. In the case of the third model of $w = 0.5$, which performs the two tasks in the most balanced way, we can see that the model design maintains all of these features. Though there exists a storage part in the middle influenced by the shaking environment, it has a tendency to minimize the water remaining in the pot through the narrowing structure from the bottom to the top which resembles the design of the pouring environment.

4.4   Contribution in Design Process

Since the computer-designed products from this experiment does not consider either usability or aesthetics other than the given objectives, it will be necessary to design a creative part based on the generated form. Therefore, in order to verify the validity of the current methodology, we invited a product designer to sketch designs through the computer-generated design. The tester chose the main design concept as a Chinese pot. He sketched the new Chinese pot using the characteristics of the generated form. The design
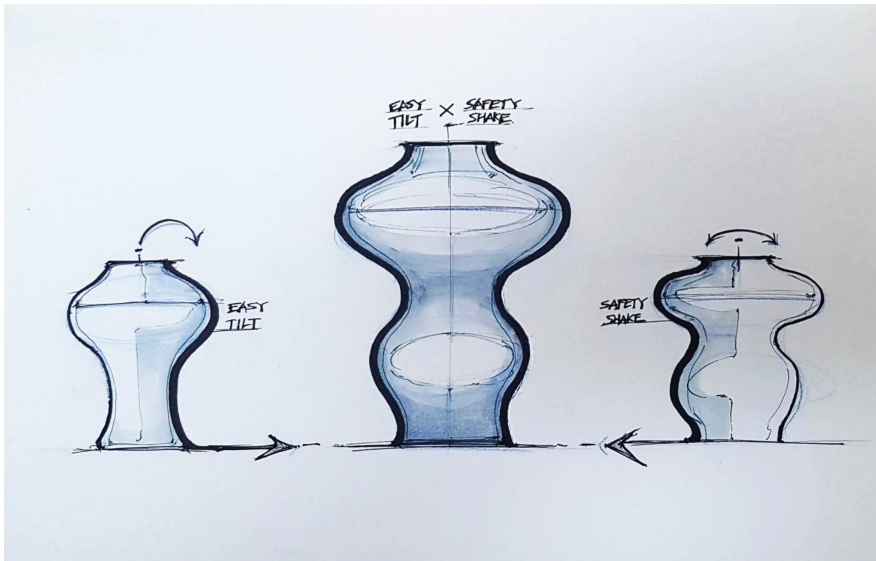
Figure 10: The sketch of a pot based on the computer-generated form. The main concept of design was Chinese pot. While designing, the product designer get a reference about the morphological concept which is applied to his actual product design concept.

sketch result is shown in Figure 10.

After sketching session, we had a short interview with the product designer to get pros and cons about this methodological concept. The designer commented that *'this methodology is highly useful when designers should make a product from a task-based concept'*. Also, he found out that *the output form is quite similar to the common sense of the pot designed for similar tasks*. As so, he believes that the results of this study will be a good reference as a task-based study, which will increase the reliability of the results produced by the designer. However, he has worried that significant features of the output can be lost because of the extreme morphological tendency of the computer-generated design.

Chapter 5

Conclusion

5.1 Conclusions

Through this study, we have shown that task-oriented design using deep reinforcement learning is possible for a specific task whose objective can be well defined. It is shown that the task objective function of the modified design through the deep reinforcement learning is significantly higher than that of the basic form, which indicates that the computer succeeded in designing the task-oriented model. By using this methodology, designers and researchers will be able to apply task-based form research before they move to creative parts of product design process. In addition, the proposed methodology is highly efficient, because it is possible to study morphology within 20 hours at low-cost, achieving a high understanding of the task.

However, in the present learning, since an action space is used in which a radius of each point layer is simply changed, there is a limit to an aesthetic or complex design that can be used in real life as an output. Also, since the reward function for the task is simply designed, the limit is shown when the performance reaches a certain level. For this reason, we will need to design

a more delicate action space as well as a reward function in future works.

Bibliography

[1] Ilpo Koskinen, John Zimmerman, T.B.J.R.S.W.: Design Research through Practice from the lab, field and showroom. Morgan Kaufmann (2011)

[2] Dorst, K., Cross, N.: Creativity in the design process: co-evolution of problem–solution. Design Studies **22** (2001) 425–437

[3] Ullman, D.G.: The Mechanical Design Process. McGraw-Hill Education, New York (1991)

[4] Aish, R., Stam, J., Glueck, M., Khan, A.: Physics-based generative design. CAAD Futures Conference **14** (2009) 231–244

[5] Du, T., Schulz, A., Zhu, B., Bickel, B., Matusik, W.: Computational multicopter design. ACM Transactions on Graphics **35**(227) (2016)

[6] Parish, Y.I.H., Müller, P.: Procedural modeling of cities. SIGGRAPH **8** (2001)

[7] Umentani, N., Igarashi, T., Mitra, N.J.: Guided exploration of physically valid shapes for furniture design. ACM Transactions on Graphics **31**(86) (2012)

[8] Zhu, Y., Zhao, Y., Zhu, S.C.: Understanding tools: Task-oriented object modeling, learning and recognition. CVPR **10** (2015)

[9] Kazi, R.H., Grossman, T., Cheong, H., Hashemi, A., Fitzmaurice, G.: Dreamsketch: Early stage 3d design explorations with sketching and generative design. UIST **14** (2017) 401–414

[10] Pahn, F., Senin, N., Wallace, D.: Distribution modeling and evaluation of product design problems. Computer-Aided Design **30** (1998) 411–423

[11] Blender Online Community: Blender - a 3D modelling and rendering package. Blender Foundation, Blender Institute, Amsterdam. (2018)

[12] Bang, A.L., Krogh, P.G., Ludvigsen, M., Markussen, T.: The role of hypothesis in constructive design research. The Art of Research IV **11** (2012)

[13] Sutton, R., Barto, A.: Reinforcement learning: An introduction, (complete draft) (2017)

[14] Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M.: Deterministic policy gradient algorithms. In: ICML. (2014)

[15] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)

[16] Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)

[17] Peters, J., Schaal, S.: Natural actor-critic. Neurocomputing **71**(7-9) (2008) 1180–1190

[18] Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P.: Trust region policy optimization. In: International Conference on Machine Learning. (2015) 1889–1897

[19] Kullback, S., Leibler, R.A.: On information and sufficiency. The annals of mathematical statistics **22**(1) (1951) 79–86

[20] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)

[21] Bellemare, M.G., Naddaf, Y., Veness, J., Bowling, M.: The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research **47** (2013) 253–279

Appendix

Experiment Result Details

In this section, we'll show the detailed rendering view of our experiment in section 4. With blender, we could simulate and render the result as video files, which can directly and intuitively see how our model works in given tasks.

As we see in figure 11, initial state of pouring environments spills a lot of fluid because of the distance between cup and the pot. In the other hand, (B) in figure 11 shows that the designed pot have inner structure that can can accelerate the fluid velocity so that it can spill less water compared to the (A). In (C) and (D) in figure 12, we can see massive improvement on shaking task. Compared to the (C), which is initial state on shaking environment, the inner structure of (D) effectively keep water in the pot.

Figure 12 represent the result of hybrid learning shown in section 4.3. Each form of pot contains the key feature of pots designed in each environment. When coefficient is 0.1m, as shown in (A) and (D), pot spills a lot of water on pouring environment, but is very good at keeping water in shaking environment. In the other hand, (C) and (F) shows the opposite tendency. (B) and (E) shows the balanced design for both environment. As we render this images, we can analyze the movement of fluid and the form of designed pot.
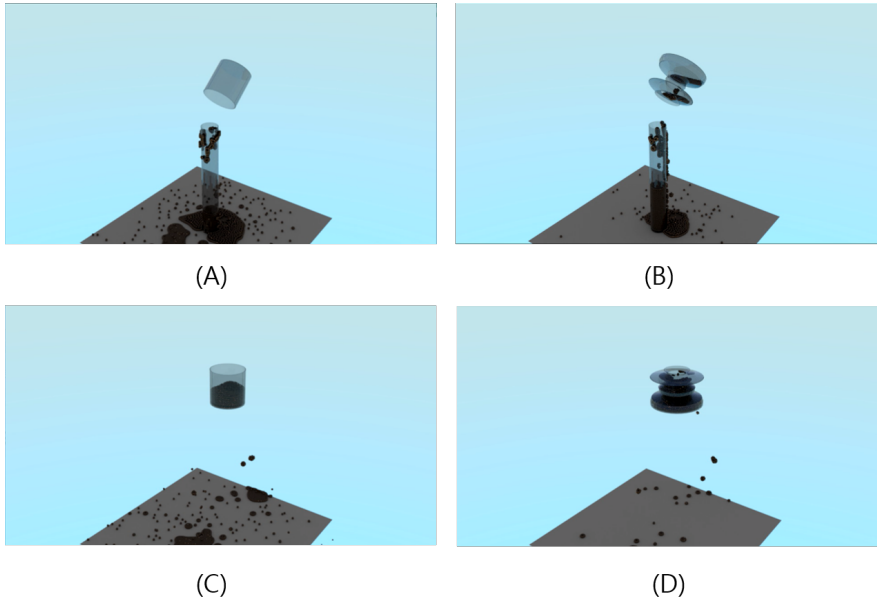
Figure 11: Rendered image of pouring and shaking environment on Section 4.1, 4.2. (A) and (B) shows the result on pouring environment. While (A) shows the initial state of learning which spilled a lot of waters on the outside of the cup, (B), the highest score on pouring environment, shows pretty good result on pouring task compared to (A). Also, (C) and (D) shows the result on shaking environments. On initial state, cup spilled nearly 60% of water, but after the learning process, it kept 86% of water.

(A) coef=0.1, pouring   (B) coef=0.5, pouring   (C) coef=0.9, pouring

(D) coef=0.1, shaking   (E) coef=0.5, shaking   (F) coef=0.9, shaking
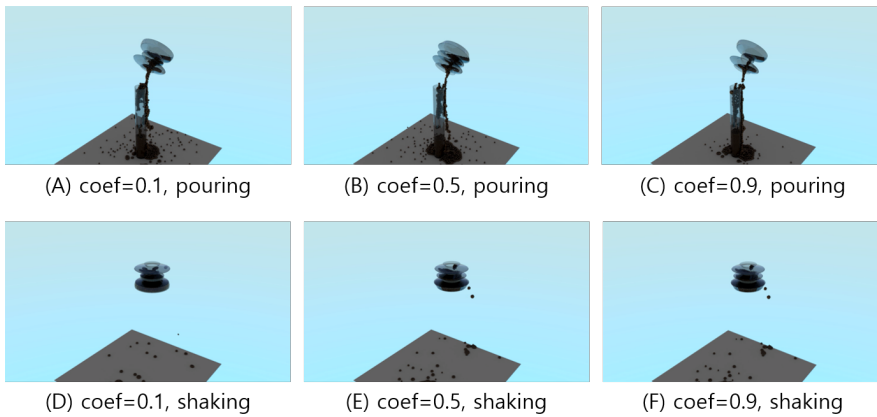
Figure 12: Rendered image of pouring and shaking environment on Section 4.1, 4.2. (A) and (B) shows the result on pouring environment. While (A) shows the initial state of learning which spilled a lot of waters on the outside of the cup, (B), the highest score on pouring environment, shows pretty good result on pouring task compared to (A). Also, (C) and (D) shows the result on shaking environments. On initial state, cup spilled nearly 60% of water, but after the learning process, it kept 86% of water.

요 약

심층 강화학습을 이용한
과제 기반의 디자인 방법론

    본 논문에서는 디자이너가 디자인 프로세스에서 문제와 솔루션 간의 갭을 줄이는데 도움이 되는 새로운 저비용 강화학습 기반의 새로운 방법론을 제시한다. 우리의 작업은 강화 학습 (RL)을 적용하여 각 과제(task)에 대한 디자인적 액션 공간(action space)을 구성하여 최적의 과제 중심 설계 솔루션을 찾는다. 이 과제 중심 설계에서는 제품 설계의 3D 설계 프로세스가 Deep RL의 액션 공간에 할당되고 과제에 따라 각 디자인 작업을 학습하여 각 과제에 알맞는 3D 디자인을 얻어낸다. 본 논문을 통해 제시되는 방법론은 또한, 여러 가지 과제를 동시에 수행해야하는 경우에도 두 개의 과제에 대해 만족스러운 결과를 내는 디자인을 얻어냄으로써 강화학습이 디자인 과정에 어떻게 기여할 수 있는지에 대한 지침을 제시한다. 또한 제품 디자이너와의 협업을 통해 본 방법론이 디자인 과정에서의 창의적인 부분에서 도움을 줄 수 있을 것이라 확인하였다.

주요어 : 과제 기반의 디자인, 머신러닝을 이용한 디자인, 강화학습, 딥러닝

학번 : 2017-22087