Improvised Robotic Design with Found Objects

Azumi Maekawa[1]*, **Ayaka Kume**[2], **Hironori Yoshida**[2], **Jun Hatori**[2],
Jason Naradowsky[2], **Shunta Saito**[2]
[1]University of Tokyo
[2]Preferred Networks, Inc.
`{kume, hyoshida, hatori, narad, shunta}@preferred.jp`

Abstract

We present a study in the creative design of robots using found objects. In particular, this study focuses on learning locomotion techniques for robots with nontraditional and arbitrary shaped limbs, in this case, tree branches. Through a use of 3D scanning, simulation, and deep reinforcement learning, we show that we can effectively learn movement strategies for such robots with unorthodox shapes.

1 Introduction

Under the banner of "found object" or "readymade", artists can give new meaning to ordinary objects. Duchamp famously placed a bicycle fork and wheel upside-down on a wooden stool, where he could appreciate it purely for its aesthetic value. Goldsworthy assigns new meaning to natural materials, such as stones and leaves, through their purposeful and aesthetic arrangement. Though these materials are mundane, the intent gives them artistic meaning.

In this paper, we consider the use of found objects in robotics. Here, these are branches of various shapes. Such objects have been used in art or architecture but not normally considered as robotic materials [5, 3]. When the robot is trained towards the goal of efficient locomotion, these parts adopt new meaning: hopping legs, dragging arms, spinning hips, or yet unnamed creative mechanisms of propulsion. Importantly, these learned strategies, and thus the meanings we might assign to such found object parts, are a product of optimization and not known prior to learning.

We propose 1) a new concept for robot design using found objects (Fig. 1), 2) a method of locomotion search using deep RL for robots consisting of irregularly-shaped found objects, and 3) an experimental case study constructing a real robot from tree branches. This opens up the door for more creative robot design, where combinations of household items or natural objects can be coordinated to perform simple tasks.

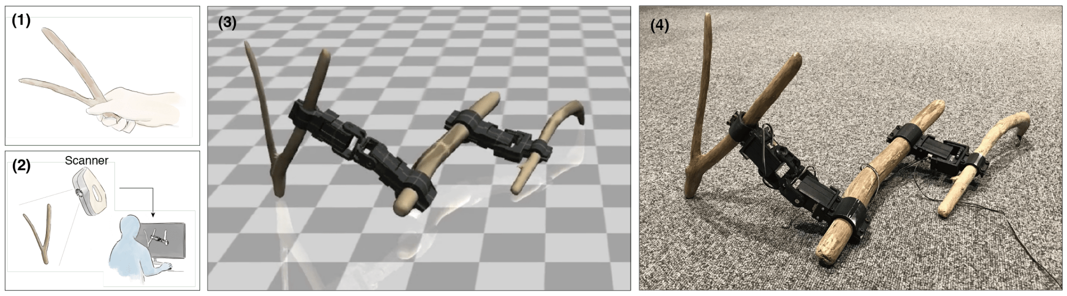

Figure 1: (1) artists pick up branches, (2) scanning and design a configuration of a robot, (3) acquire effective locomotion policy in a simulator, (4) transfer to the real robot.

*Work performed while the author was doing an internship at Preferred Networks

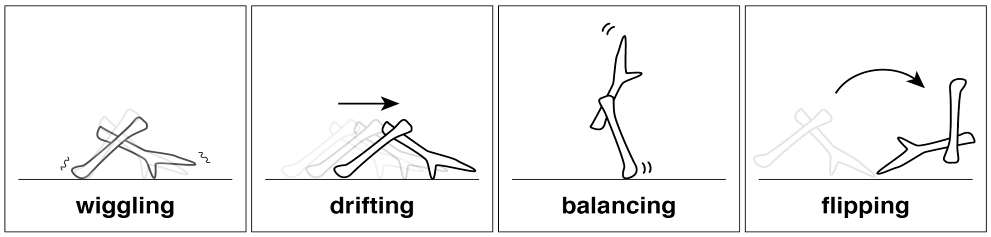

Figure 2: Four major invalid states observed in simulation.

2 Our robot

We collected various Y or I shaped branches to serve as the robot body, and connected these to servo motors to complete the robot. Having designed our robot, we turn to the problem of assigning function to form. We scanned each branch with a 3D scanner, and simplified the scanned geometries for more efficient simulation. Weights were measured, and the values of the moment of inertia values were geometrically calculated. By scanning such objects and incorporating them into a digital robot model, we can perform learning in a simulated environment.

3 Learning

Effective Locomotion In the simulator, we allow the model to explore various movements and, via reinforcement learning, learn to apply them towards the goal of effective locomotion. We use the DDPG (Deep Deterministic Policy Gradient) [4], which is a model-free algorithm that can be used with continuous multi-dimensional action spaces, like in servo control. DDPG has been used in prior works with real robots [2]. Here learning in the simulator is advantageous, as it allows for multi-agents learning, and a large number of associated failure attempts, without destroying the physical robot.

Reward Shaping We can control the learned policy indirectly by specifying different reward functions. For example, an agent gets greater reward when it gets far from the initial position, so that the agent will learn an efficient gait to go far after training. We can also steer the model away from unfavorable policies (Fig. 2) by giving low rewards for specific behaviors, such as spinning at the same position, or other motions that might cause stress and wear in the real robot.We discuss this further in the Supplementary Materials as these conditions are closely related to the hardware setup of the real robot.

Transferring from simulation to reality To apply the learned motion to the real robot, we simply copy the joint values from the simulator policy to the servo motors on the real robot. While fine-tuning the policy in the real world environment is one option, we manually select policies which satisfy both high expected returns and high feasibility in the real world environment (as judged by human observation of the simulation). In the end, we collected 30 different policies and curated them based on effectiveness and aesthetic value. We present one in supplementary video[2], chosen for its unique locomotion using a Y-shaped branch.

4 Lessons Learned

In this paper, we proposed a new concept of creating a robot using irregularly-shaped found objects. It also revealed several remaining challenges, such as the need to further overcome the gaps between simulator and the real world. But in many ways we were successful, and achieved the goal of creating real-world robots that can utilize their unorthodox bodies in unorthodox ways. While none of the learned behaviors are refined enough to be considered walking, their crawling behaviors often mimic those found in animals, and in doing so we assign new meaning and purpose to commonplace objects.

[2]`https://drive.google.com/open?id=1TYpNKAicftVz3gLZPk8i3X1bhJSN3WYI`

References

[1] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[2] Sehoon Ha, Joohyung Kim, and Katsu Yamane. Automated deep reinforcement learning environment for hardware of a modular legged robot. In *15th International Conference on Ubiquitous Robots, UR 2018, Honolulu, HI, USA, June 26-30, 2018*, pages 348–354, 2018.

[3] Takeo Igarashi Hironori Yoshida, Maria Larsson. Upcycling tree branches as architectural elements through crowdsourced design and fabrication, 2019.

[4] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[5] Jessica Mairs. AA design and make students use a robotic arm to build a woodland barn. `http://pr2016.aaschool.ac.uk/2014-15--Woodchip-Barn`, 2016. Accessed: 2018-07-30.

[6] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[7] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

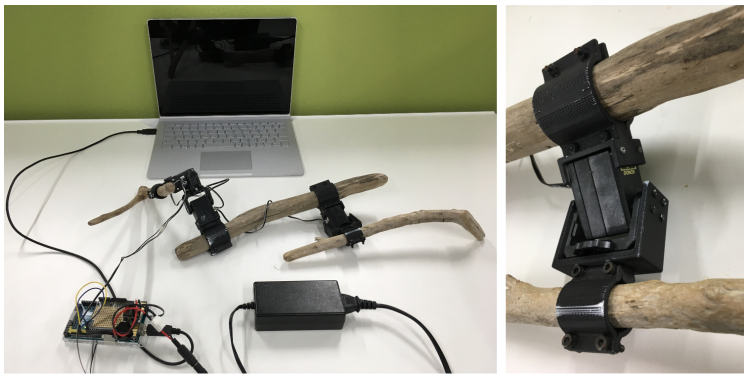
Figure 3: A robot constructed as a case study. Left: The branch robot is controlled by Arduino Mega 2560 and a motor driver with a separate power supply. The size and weights of the three branches (left, middle, and right) are 339, 383, 271 (mm), and 56, 120, 38 (g) respectively. Right: A close-up of the connector. Kondo KRS-2572HV servo motor is connected with branches.

Supplementary Materials

A Case Study

The goal of this case study is to verify the feasibility of the proposed work-flow which is starting from designing the robot, learning gait in a simulator, and verification of the gait with a real robot.

A.1 Hardware setup

We choose tree branches as hardware material of the robot. Tree branches have varying shapes, and their rigid and lightweight properties are suitable for a physics-based simulation. Most importantly, tree branches are found everywhere thus they deserve to be called found objects.

We collected twenty branches and used six of them for this case study. The composition of the branches is manually defined (see Figure 1.4). We designed two types of robots each of which consists of three branches: one has three motors, while the other has four (see the left picture in Figure 3). The branches and servo motors are connected by connectors printed by 3D printer (see the right picture in Figure 3). We scanned each branch with Artec EVA hand-held 3D scanner, and then simplified the scanned geometries for light-weight simulation. Weights were measured, and the values of the moment of inertia values were geometrically calculated.

A.2 Simulation setup

We used MuJoCo [7] which is a standard simulator used in OpenAI gym [1]. The simulation ran on a Linux server which has a Intel Xeon E5-2623 v3 CPU and a NVIDIA GeForce GTX TITAN X (Maxwell). We empirically chose 0.5 for the friction coefficient of the floor on the simulation, and ran the simulation at 10 Hz to synchronize it with the real robot. Figure 4 describes the task setup for the simulation.

A.3 Reward functions

To obtain feasible gaits for the real robot, the reward function was needed to be carefully designed because some learned gaits which work in the simulation did not work with the real robot. For example, we found that joint values tended to be either minimum or maximum values ($\pm 135°$) at each time step during training, resulting in wiggling motion. The wiggling motion is not only inefficient, but also infeasible with a real robot. Another simulation-specific gait was drifting (or sliding). We avoided this by giving a negative reward when the difference between succeeding actions is small. Flipping motion and balancing action are also problematic. Flipping causes the entanglement of wires in the real world. We avoid balancing case as it is sensitive with small difference between simulation and the real world. As flipping and balancing were not frequently observed, we manually

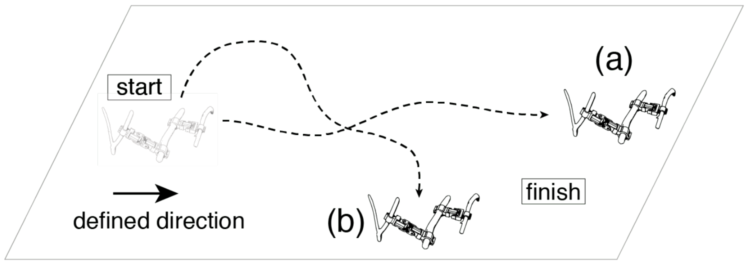

Figure 4: Simulation setup for RL. The robot receives high reward as it moves straight to the defined direction (a), otherwise, low reward due to out of the path (b).

filtered these out by checking the rendered frames. Figure 2 summarizes these problematic gaits. Please refer to the supplementary video material for these gaits.

A.4 Training of the agent

Each agent took few hours to finish learning, which is approximately a few hundred thousand time steps in simulator. The neural network used for both actor and critic in DDPG had two hidden fully connected layers with 100 hidden units. During the learning, we prepared 30 agents with varying reward functions for a diverse set of gaits, as well as to avoid the above mentioned invalid gaits. Further, other DRL methods, such as Deep Q-Learning [6], were tested in this step.

A.5 Application of the learned policy to the real robot

We sorted agents based on scores, and then selected three agents by manually checking rendered frames. Some invalid gaits, such as flipping and balancing, were filtered out in this step.

One of agents acquired a propelling motion with Y shaped branches (see Figure 1.3 and 4), which was an unexpected gait for the authors. This motion was recorded in the simulator, and then its joint values were exported for application to the real robot. As a result, we observed that the robot successfully moved forward as in the simulator without flipping and wiggling, nor did it remain stacked in the same spot. Please refer to the video material for the motion of the actual robot.

A.6 Remaining Challenges

Although we managed to complete the work-flow, this case study revealed several challenges. First of all, it took several iterations, going back-and-forth, of the work-flow to minimize the gap between the simulator and the actual robot. For example, we had to update the friction coefficient of the floor and the scale of rewards several times. Similarly, the mesh size of the object is supposed to influence the the simulation result, particularly in contact calculation; however, we did not iterate trial-and-error on this parameter. There could be other parameters we did not pay attention to in this case study.

Ideally, the above points could be solved by running the learning with the real robot; however, there are several challenges described as follows. In fact, we tried to prepare the DRL setup with the real world; however, it did not obtain sufficient results. The main reason for this was that the duration of the learning process would be very long. Observation of the pose was also an issue due to the relatively small size of the branches. If the size were larger, it would be possible to put cameras on the robot itself, which would reduce the gap between the simulation and real-world setup.

A larger size is also suitable for making the robot stand-alone without the external motor controller and power sources, which mitigates the flipping constraint. During the case study, we observed an entanglement of the cable several times, even with the gait moving straight, without spinning or rotating.