On Using Wave Variables for Robot Imitation Learning

Phongsaen Pitakwatchara

Abstract— **There are several options in specifying the action for robot control. Typically, robot motion is the natural choice since it is a readily observable quantity. However, applied force may be more suitable for tasks that involve contact with the environment. Moreover, many tasks that exhaustively interact with the environment typically require a specification of both motion and force simultaneously. The wave variable combines them into one quantity related to the power supplied to the robot. Therefore, wave variable may be used to specify the action. This helps the robot perform interactive tasks better than using motion or force alone. It also helps in creating high quality dataset of task demonstration since human perceives wave reaction as an additional modality to generate the wave action properly.**

I. Introduction

Robots have been successfully used to perform challenging manipulation tasks in the real world thanks to the help of AI technology. In particular, neural networks have been used to determine some intermediate quantities, such as the position of the object to grasp or the motion of the robot at the next time step. They will be sent to the low-level controller that will compute the torques for the robot. This approach is known as the modular approach.

Some works along this line are as follows. A deep neural network was trained in [1]–[3] to predict the Cartesian motion of the gripper toward the target object for grasping. A video prediction model [4] was used together with the trajectory optimizer to compute the best relative movement of the end-effector in performing general manipulation tasks. A robot hand was developed and controlled [5] by a policy network that outputs a discretized joint angle displacement as the action.

Similarly, a policy network was trained to predict the target joint angles of a humanoid robot [6], of which the torques are computed by the PD controller. In addition to the position controller, parallel position/force and admittance controller [7], and impedance controller [8] were used in conjunction with the network that output the pose command and the controller parameters as its action. Recently, human stiffness was explicitly learned and transferred to the teleoperation system by adjusting its controller stiffness [9]. A high-level physics-represented discrete action space was proposed and learned for tasks performed with the Shadow hand [10].

There are some issues related to the interplay of the neural network and the low-level controller systems. The network will be relatively easy to train if it is used to infer some quantities on top of the raw data sensor, e.g. position of the object from the image frame. For the controller, it must be robust to the output errors of these networks. In addition, a rule-based system may be needed that provides high-level logic and commands to orchestrate various modules and information for each stage.

On the other hand, a deep neural network could be used to determine the joint torques of the robot from the input stream of the sensors, which will then be commanded directly. This is the end-to-end approach. For example, an end-to-end policy was trained, which observes the stream of images and robot configuration data and outputs the joint torques [11] using reinforcement learning and a guided policy search algorithm. In [12], the locomotion of a quadruped was controlled by a torque action policy network optimized using the proximal policy optimization algorithm. This approach eliminates the need for the controller and the rule-based systems in exchange for more complex network and the difficulty in training.

It is apparent that many works have employed the neural network with robot motion as its action space. The stream of the predicted output will be given as the desired trajectory to the motion-input controller. The impedance controller is usually preferred because of its simplicity and robustness to uncertainties. With this choice, the neural network is quite easier to train compared to the one having force or torque as its action space. The success rates of the tasks dominated by the movement are also higher.

Still, there are many tasks which require the robot to interact with the environment intensively such that a significant order of power flows among them, for example wood carving or wiping a dirty table. Specifying either the robot motion or the applied force to the underlying controller might not address all aspects of the manipulation and will likely lead to unsuccessful work. A hybrid controller may be adopted, but it requires specifying the robot motion and the applied force commands in the proper direction at each time step. In turn, the action space of the network must be extended and the learning will be much more difficult.

These issues inspire us to develop a method to specify the power as an action for robot control. Essentially, the contribution of this paper is the neural network which is trained to predict the power command, instead of the motion or force, necessary to drive the robot to perform the task. The power will be sent to the controller to compute the joint torques of the robot.

We developed the wave planner framework [13] that describes such an idea using wave input and output to represent

The author is with the Department of Mechanical Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok, 10330, Thailand (e-mail:phongsaen.p@chula.ac.th). This work was supported in part by the grants from Faculty of Engineering and from Thailand Science Research and Innovation Fund, Chulalongkorn University (IND_FF_68_045_2100_007).

the power flow. In this work, we apply the framework with the input wave being inferred from the trained neural network. Simulation shows that the proposed wave action control architecture can be successfully applied to perform arbitrary robot manipulation tasks through imitation learning. However, those using robot motion as the action space fail for tasks that highly interact with the environment.

The outline of the paper is as follows. Section II briefly reviews the wave variables. The collection of dataset, which also include wave action labels, and the proposed control architecture are described in Section III. In section IV, we present and discuss the simulations of various controllers for different tasks. Section V concludes the paper with the findings.

II. Wave Variables

Wave variables, which consist of input and output waves denoted as u and v, are an alternative representation of the power variables, e.g. force f and motion $\dot{x}$ at a port of interaction in the mechanical energy domain. They are defined as

$$u \equiv \frac{1}{\sqrt{2b}}(b\dot{x} + f) \qquad v \equiv \frac{1}{\sqrt{2b}}(b\dot{x} - f) \tag{1}$$

where the adjustable parameter b is the wave impedance with the unit of [N·s/m] that weighs the combination of f and $\dot{x}$. Equation (1) may be rearranged in different ways that allow any two variables to be expressed in terms of the other two variables.

Wave variables are closely related to the power flowing through the port since the total power flow P may be expressed as

$$P = f^\top \dot{x} = \frac{1}{2}(u^\top u - v^\top v) \tag{2}$$

It is seen that the unit of wave variable is $[\sqrt{\text{Watt}}]$. Thus, u and v signify the power flowing into and out of the port respectively, constituting the net power flow.

Wave variable may be used to command robot manipulation, as it aligns with the intuitive notion of exchanging power between the robot and the environment. For general manipulation, it is unclear whether a person is executing the desired force or motion. Nevertheless, it is certain that he is providing power to the environment. Therefore, we will apply this notion to the robot, i.e. supply the power to the robot such that it interacts with the environment properly. See Fig. 1 for the net power flow from the controller to the environment.

III. Imitation Learning Using Wave Variables

Currently, there are two major trends in robot learning: imitation and reinforcement learning. Imitation learning (IL) [14] attempts to mimic the behavior of a person in performing the task from the collected demonstration dataset. In contrast, reinforcement learning (RL) [15] in its basic form does not require demonstration data. It attempts to find the optimal action policy that will maximize the rewards given by repeatedly interacting with the environment and improving the policy.

We will use the IL approach in this work. Specifically, a person will control the robot to perform the task based on what he perceives and feels. Relevant data on robot pose and motion (x_e and $\dot{x}_e$), interaction force (f_e), environmental states (s), and wave action and reaction (u_r and v_r), may be collected for each demonstration as the dataset for learning.

A. Collection of dataset

We propose using the wave telerobotic framework in [16] as a method to collect data for training the deep neural network to predict suitable wave action. Specifically, the operator manipulates the master robot which connects to the slave to perform the task. Since the real robots are not available in this work, we use a SpaceMouse as the master to deduce the motion of the operator and a simulated Franka arm in MuJoCo as the slave. However, it should be noted that the use of real robots allows the operator to physically sense the wave reaction feedback, which provides an additional modality for the operator to generate the wave action properly. This should improve the quality of the current dataset.

The robots are connected through the bilateral transmission of wave variables. According to the wave robotic framework of Fig. 2 in [16], the input wave from the master to the slave, $\hat{u}_s$, may serve as the label for the neural network that predicts wave action. Robot pose, environmental states, previous input and output waves are the observed data recorded accordingly. High-dimensional data, such as images, could also be included.

B. Neural network model

A deep neural network parameterized by θ serves as the robot policy for a specific task. It receives the stream of observed data o as input, and sequentially predicts the wave action u_r at the controller input port r as output;

$$u_r = \pi_\theta(o) = \pi_\theta(x_e, s, u_r^-, v_r^-) \tag{3}$$

The observed data basically consist of the robot pose and the environmental states. Additional data on previous wave action and reaction are observed for the network to predict the next wave action. Moreover, we examine what would happen when the interaction force has also been observed.

These low-dimensional data modalities are simply concatenated into a single long vector and feed as input to a MLP model. An output vector of the prescribed size is then computed, which is then passed to the transformer model. The output of the transformer at the last layer is a representation vector that embeds useful information about the next action. It is passed through another MLP model that outputs the predicted action accordingly.

We leverage the standard transformer [22] as the underlying architecture, since it is an expressive deep neural network model that can address sequential data well. The loss function for training is simply the mean square error (MSE) of the predicted actions

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}\|\hat{u}_s^{(i)} - u_r^{(i)}\|^2 \tag{4}$$

for the batch of data size N. We use the cosine annealing scheduler with warm restarts [23] to adaptively set the learning rate from 1×10^{-4} down to 1×10^{-5}, on every 100 epochs. This effectively avoids the problem of local minima. Hyper-parameters of the network, described in section IV, are chosen to suit each specific task.

C. Low-level controller

In contrast to predicting motion or force directly, the network may be trained to predict the wave action, which is then used to determine the reference motion or force based on the input type of the low-level controller [13]. In this work, we apply a simple impedance controller [18] at the robot interaction port e, with the damping and spring constants, c and k, of the form

$$f_r = c(\dot{x}_r - \dot{x}_e) + k(x_r - x_e) + d\dot{x}_r \tag{5}$$

Additional damping $d\dot{x}_r$ is for reducing the wave reflection [17]. Since the controller requires the reference motion $\dot{x}_r$ as input, it will be calculated from the companion force f_r and u_r according to (1) as

$$\dot{x}_r = \sqrt{\frac{2}{b}}u_r - \frac{f_r}{b} \tag{6}$$

The proposed "WaveAction" robot control architecture is summarized in Fig. 1. The neural network (3) is connected to the impedance controller (5) using (6) to determine the reference motion from wave action and reaction force while preserving the power. This is fundamentally different from previous works [1]–[3], [5]–[8] having the neural networks predict $\dot{x}_r$ and send it directly to the controller. In this work, $\dot{x}_r$ is deduced from the wave action u_r related to the input power, which is a unified and enriched quantity for general tasks. Specifying the wave action yields superior result than the motion, especially for interactive tasks or tasks involving contact with the environment in which it is difficult to determine proper reference motion directly. Furthermore, $\dot{x}_r$ is corrected on the fly by the reaction force f_r. Thus, harmful motion that leads to large reaction force will also be instantly mitigated.

Other motion-input controllers may be adopted in place of the simple impedance controller (5). For example, we also experiment with the operational space controller [19]

$$f_r = \widehat{M}(x_e)\left[\ddot{x}_r + k_v(\dot{x}_r - \dot{x}_e) + k_p(x_r - x_e)\right] + d\dot{x}_r \tag{7}$$

where $\widehat{M}(x_e)$ is the estimated robot inertia matrix in the end-effector space. This controller essentially cancels nonlinear dynamics of the robot motion (not shown in (7) for clarity) and adds a simple position controller to the shaped unit mass robot. If the force-input controller is used, the reference force f_r may be determined by

$$f_r = \sqrt{2b}u_r - b\dot{x}_e \tag{8}$$

IV. Simulations

We have performed extensive simulations to assess different robot control architectures for several tasks based on robosuite and robomimic projects [20], [21]. The Franka robot arm is used to perform three tasks of picking up a cube, opening a spring-latched door, and wiping a dirty table, as shown in Fig. 2. They represent the tasks that have degrees of complex interaction with the environment from low to high. The number of demonstrations for each task is 200, 300, and 500 in order. Codes for implementing the "WaveAction" control architecture and video clips are available.[1]

Typically, it requires significant amount of time for data processing and network inferencing compared to the controller computation time. To simulate this effect, we program the network to compute the action at a rate of 20 Hz while the controller and MuJoCo run at 500 Hz. Various robot control architectures have been employed in performing the tasks. We compare the proposed "WaveAction" (WA) control architecture with the "MotionAction" (MA), which is a typical one where the network predicts the reference motion and sends it to the impedance controller. We also apply the reference motion to the operational space controller, called "WaveActionOSC" (WA_OSC), i.e. using (7) instead of (5). In addition, due to large time delay in the network output update, we modify the original WA by locally interpolating the wave action to smooth out the value. This is denoted as "WaveActionINT" (WA_INT). Note that the predicted motion of MA has been interpolated by default. The controller parameters such as stiffness and damping are tuned for each task by cursory and used in the demonstration, training, and testing phases.

We employ the transformer model to decode the action from the observed data with the context length of up to 60. This implies utilizing the past information up to 3 seconds to determine the action for the next step. We design the transformer to have 4/6/8 layers, 8/8/12 heads, and embedding dimensions of 256/512/768 for the tasks of picking up a cube, opening a door, and wiping a table. These choices of hyper-parameters reflect the increasing complexity of the tasks.

The network is trained using the collected dataset from human demonstration, where 10% of it is reserved for validation. For each epoch, we step through batched training data repeatedly for about 3 times, then run the updated model on the validation data. At every 50 epochs, we roll out or test the policy 10 times with the newly randomized environments. We stop training once the success rate of the rollout exceeds 0.9. An exception is for the most difficult wiping task, where we lower the success rate down to 0.5. The obtained model is then evaluated against a set of 50 fixed environments, of which the mean and sample standard deviation of the success rate metric is computed across 10 different seed values. All trainings and evaluations are performed on a machine equipped with an Intel i7-12700F and a NVIDIA RTX 3090.

[1]`https://github.com/phongsaen/waveaction`

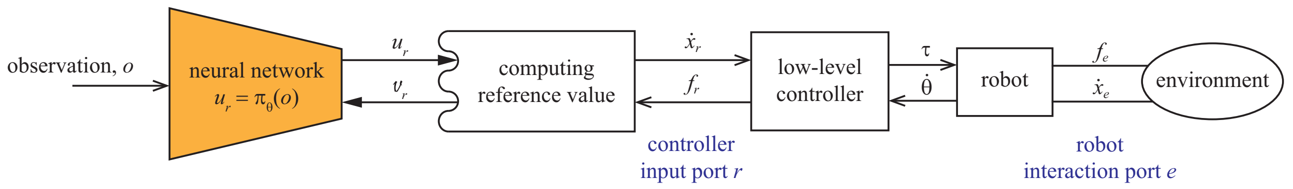

Fig. 1. "WaveAction" robot control architecture. Robot manipulation may be viewed as the supply of power to the robot in order to interact with the environment.

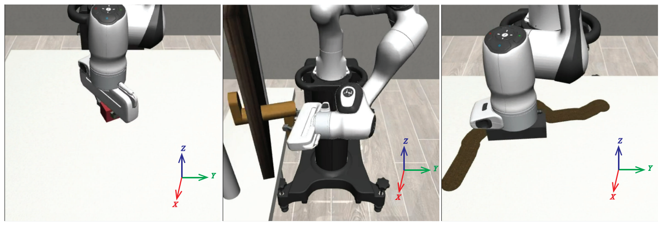

Fig. 2. Three tasks that are used to evaluate different control architectures: pick up a cube, open a spring-latched door, and wipe a dirty table.

Arch.	MA	WA	WA_OSC	WA_INT
mean	0.964	0.83	0.87	0.89
s.d.	0.0310	0.0392	0.0435	0.0474

TABLE I
SUCCESS RATE OF THE TASK "PICKING UP A CUBE".

A. Pick up a cube

A cube of varying size in the range of $[2.0, 2.2]$ cm. is placed randomly on a table with ± 3 cm. uniform position variations along the x and y-directions, and arbitrary orientation. The robot arm needs to pick the cube up to 4 cm. over the table top. The environmental states observed are the cube pose and the position from the gripper to the cube. This task is the simplest among all since the manipulation mostly involves the pure motion of the robot. Only a small amount of force is needed to lift the cube.

Table I shows the success rate of the task using different control architectures. The MA architecture has higher success rate than others. This is because, for most of the time, the robot moves freely and aligns its gripper with the cube. At the end of the task, the lightweight cube is picked up, which involves a small amount of lifting force. Thus, the applied force may be ignored and only the motion action command is sufficient. If the WA architecture is used, the neural network needs to comprehend the relationship between the dominant motion and the minuscule force to specify appropriate power to the robot. This is difficult to achieve using a small dataset.

B. Open a spring-latched door

A small-sized spring-latched door is placed randomly on a table with uniform coordinate variations in x and $y = 2$ cm. and $\theta = 0.25$ rad. The door is closed initially. The robot needs to open the door by rotating the handle, which is loaded by the winding spring of 1 N·m/rad and subject to the dry friction of 0.1 N·m for about 90°. Then, the handle must be held firmly and pulled circumferentially so that the door opens and swings by more than 0.3 rad. The environmental states observed are the position and orientation of the door and the handle, and their relative position to the gripper.

In this task, the robot involves both the free motion from its initial pose to grab the handle, and the constrained motion to rotate the handle and pull out the door with moderate amount of force along the admissible motion direction. Two sample plots, using the MA and WA_OSC, of the interaction force and motion at the robot end-effector are depicted in Fig. 3. They are expressed in the fixed reference frame shown in Fig. 2.

Arch.	MA_300	WA_300	WA_OSC_300	WA_INT_300
mean	0.836	0.694	0.786	0.728
s.d.	0.045	0.0566	0.0633	0.0738
Arch.	**MA_500**	**WA_500**	**WA_OSC_500**	**WA_INT_500**
mean	0.994	0.946	0.978	0.774
s.d.	0.0135	0.0353	0.0199	0.0481

TABLE II
SUCCESS RATE OF THE TASK "OPENING A DOOR".

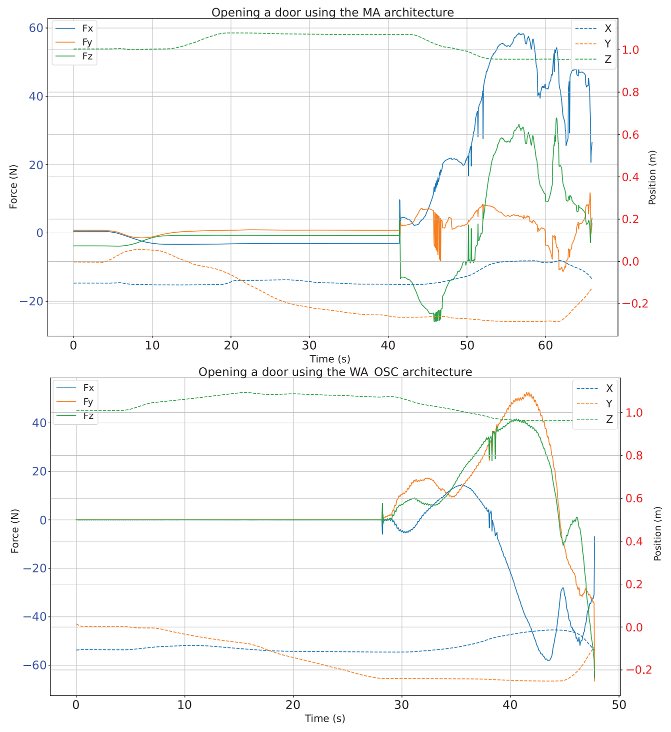

Fig. 3. Interaction force and motion at the robot end-effector during opening a door.

It is seen that during holding the handle, the force varies greatly depending on the nuances of the handle grasping and door pulling. Therefore, any control architecture having the interaction force as one of the observed data will fail to perform the tasks in general, as it is arduous for the network to comprehend the relationship between the interaction force and action involving the robot and the intricate dynamics of the environment. In contrast, the force used to determine the wave variable is the force applied to the robot by the controller. Note that the interaction force of the WA_OSC architecture fluctuates slightly due to impedance mismatch during contact. It can be eliminated by interpolating or filtering the wave action.

Table II shows the success rate of the task. The WA architecture and its variants perform worse than the MA. This is because the task of opening the door relies more on action from motion than force in both free and constrained maneuvers, such that it is immaterial how much force to be applied. In fact, it will be determined by the low-level controller. Thus, we may safely use the MA architecture.

Nevertheless, the applied force, which helps to form the wave action, is not negligible and depends heavily on the contact status and arm configuration. It must overcome the combined spring, friction, and varying inertial forces to incur the motion. This complicates the wave action, which leads to difficulty in training using a small dataset.

We thus increase the number of demonstrations from 300 to 500 and enlarge the model size accordingly. It is found that the WA and WA_OSC trained with larger dataset now perform equally well with the MA. However, success rate of the WA_INT improves just a little due to the side effect of power loss from the commanded value by interpolating the wave action.

C. *Wipe a dirty table*

A set of 100 randomly placed circular blobs, each of size 2 cm. in diameter and 2 mm. in thickness, is used to represent the dirty trace on the table. The robot needs to clean it up by wiping them using a brush mounted at the robot end-effector. The environmental states observed are the centroid of the remaining blobs, the radius of the encompassing circle centering at the centroid, the position from the brush to the centroid, the proportion of the wiped blobs so far, and the current contact status of the brush.

The task is programmed so that, during wiping, the brush surface has to penetrate the blob to be wiped out. Moreover, to emulate deep stain for some random episodes, the sliding and torsional coefficient of friction between the table and the brush are sampled from the normal distributions with the mean values of 0.03 and 0.005, and a large standard deviation of 0.3. This makes some cleaning episode very difficult to accomplish, as high friction will impede the wiping motion.

This task is the most complex one since both interaction force and motion are significant but their correlation is not apparent. Right after approaching the table, rapid contact happens, which creates an impulsive force. Then, the robot starts scrubbing the table top while exerting proper amount

Arch.	MA	WA	WA_OSC	WA_INT
mean	0.346	0.546	0.702	0.724
s.d.	0.0517	0.0574	0.0689	0.046

TABLE III

SUCCESS RATE OF THE TASK "WIPING A TABLE".

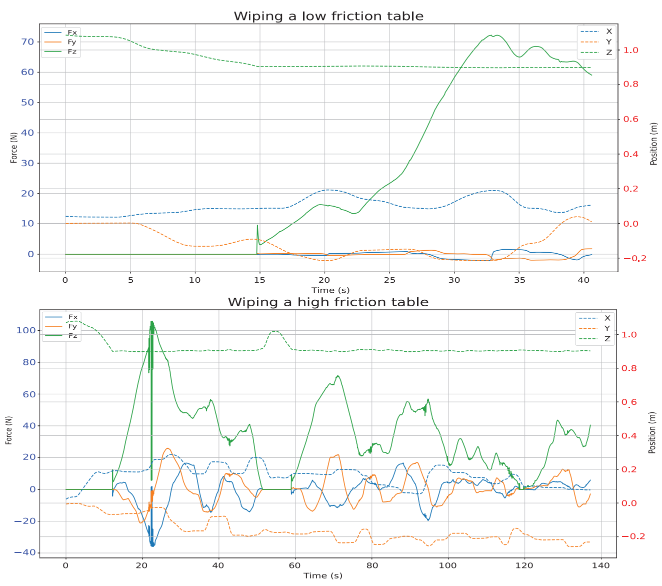

Fig. 4. Interaction force and motion at the robot end-effector during wiping.

of force in the direction normal to it. The task is even more difficult because the operator cannot perceive the reaction force through SpaceMouse.

In addition, the stain cannot be easily scrubbed out when the sliding friction is large. This is not known until we try to wipe it and see that the brush gets stuck. A viable solution for this scenario is to apply the normal force so that the brush penetrates the blobs underneath, which will then be wiped out without having to scrub it. After that, we lift up the brush, move to unwiped blobs, and re-apply the normal force.

Two sample plots of the interaction force and motion at the robot end-effector are depicted in Fig. 4. One is from wiping a low friction table and the other from high friction. In the latter case, we observe moderate friction force along the x and y-axes and large and bumpy normal force due to our strategy of pressing and releasing the brush against the table. The corresponding wave action and reaction are plotted in Fig. 5. They are nearly negative to each other because friction and normal force behave like a wall reflecting the incoming wave to the returning wave. The network is able to recognize this pattern and, together with other observed data, generate suitable wave action accordingly.

Table III shows the success rate of the task. Most failure tasks are those with a high friction value. The MA architecture has the lowest success rate, as it does not account for the force that plays an equal role to the motion in this task. Among the wave action networks, the WA_INT yields the highest rate because the smoothened wave action makes the

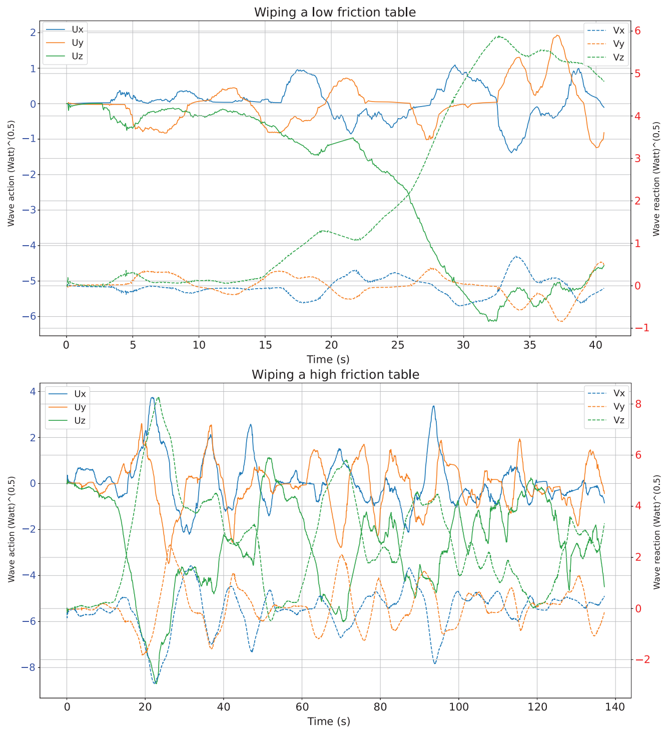

Fig. 5. Wave action and reaction at the robot end-effector during wiping.

relevant data less noisy and thus easier to learn. Its power loss does not affect much because, unlike opening the door, the operator usually overly supplies the power for wiping.

V. Conclusions

We propose the wave action control architecture for the robot to perform manipulation tasks via imitation learning. It consists of three components: the neural network which learns to predict the power required for manipulation through the wave action space, the transformation that determines the reference motion from the wave action and reaction force while respecting the supplied power, and the motion-input controller. Accordingly, we also propose to collect the wave variables and other relevant data using the wave telerobotic framework.

We compare the wave action control architecture and its variants with the typical one that determines the motion as its action space over three different tasks. It is observed that the WA control architecture requires more data and steps to train than the MA because the wave action contains richer information from both force and motion variables. After sufficiently training, both the MA and WA architectures can be used to perform tasks that have little interaction with the environment or are dominated by motion equally well. However, WA shines and yields a higher success rate for tasks that interact extensively with the environment, such as wiping a table. It seems to discover the right strategy for the task with unknown parameters at hand. We plan to improve the WA architecture further in several aspects, such as learning the adaptive wave impedance parameter online and introducing the probabilistic action, and eventually apply the method to the real robot.

References

[1] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *arXiv preprint* arXiv:1603.02199, 2016.
[2] D. Kalashnikov, et. al., "QT-Opt: scalable deep reinforcement learning for vision-based robotic manipulation," *Conf. on Robot Learn.*, 2018.
[3] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," *IEEE Int. Conf. Robot. Autom.*, pp. 5628-5635, 2018.
[4] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine, "Visual foresight: model-based deep reinforcement learning for vision-based robotic control," *arXiv preprint* arXiv:1812.00568, 2018.
[5] I. Akkaya, et. al., "Solving rubik's cube with a robot hand," *arXiv preprint* arXiv:1910.07113, 2019.
[6] X. Peng, P. Abbeel, S. Levine, and M. Panne, "DeepMimic: example-guided deep reinforcement learning of physics-based character skills," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 1-18, 2018.
[7] C. C. B.-Hernandez, D. Petit, I. G. R.-Alpizar, T. Nishi, S. Kikuchi, T. Matsubara, and K. Harada, "Learning force control for contach-rich manipulation tasks with rigid position-controlled robots," *IEEE Robot. Autom. Letters*, vol. 5, no. 4, pp. 5709-5716, 2020.
[8] R. M. Martin, M. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg, "Variable impedance control in end-effector space: an action space for reinforcement learning in contact-rich tasks," *IEEE Int. Conf. Intell. Robots Syst.*, pp. 2062-2069, 2019.
[9] L. Situ, Z. Lu, W. Si, and C. Yang, "Human multi-dimensional stiffness transfer for robot teleoperation system," *IEEE Int. Conf. Syst. Man Cybern.*, pp. 321-327, 2024.
[10] Y. Wu, M. Hu, R. Jin, and R. Liu, "Physics representation learning for dexterous manipulation planning," *IEEE Int. Conf. Robot Hum. Interact. Commun.*, pp. 1177-1182, 2024.
[11] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, no. 39, pp. 1-40, 2016.
[12] S. Chen, B. Zhang, M. Mueller, A. Rai, and K. Sreenath, "Learning torque control for quadrupedal locomotion," *IEEE-RAS Int. Conf. Human. Robots*, pp. 1-8, 2023.
[13] P. Pitakwatchara and J. Arunrat, "Manipulation planning using wave variables," *IEEE Robot. Autom. Letters*, vol. 7, no. 3, pp. 6179-6186, 2022.
[14] M. Zare, P. M. Kebria, A. Khosravi, and S. Nahavandi, "A survey of imitation learning: algorithms, recent developments, and challenges," *IEEE Trans. Cybern.*, vol. 54, no. 12, pp. 7173-7186, 2024.
[15] R. S. Sutton and A. G. Barto, "Reinforcement learning: an introduction," MA, USA: The MIT Press, 2018.
[16] P. Pitakwatchara, "Wave correction scheme for task-space control of time-varying delayed teleoperation systems," *IEEE Trans. Control Syst. Tech.*, vol. 26, no. 6, pp. 2223-2231, 2018.
[17] G. Niemeyer and J.-J. Slotine, "Designing force reflecting teleoperators with large time delays to appear as virtual tools," *IEEE Int. Conf. Robot. Autom.*, pp. 2212-2218, 1997.
[18] N. Hogan, "Impedance control: an approach to manipulation," *ASME J. Dyn. Syst., Meas., Control*, vol. 107, no. 1, pp. 1-24, 1985.
[19] O. Khatib, "A unified approach for motion and force control of robot manipulators: the operational space formulation," *IEEE J. Robots. Autom.*, vol. 3, no. 1, pp. 43-53, 1987.
[20] Y. Zhu, J. Wong, A. Mandlekar, R. Martin, A. Joshi, K. Lin, A. Maddukuri, S. Nasiriany, and Y. Zhu, "A modular simulation framework and benchmark for robot learning," *arXiv preprint* arXiv:2009.12293, 2020.
[21] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. F.-Fei, S. Savarese, Y. Zhu, and R. Martin, "What matters in learning from offline human demonstrations for robot manipulation," *Conf. on Robot Learn.*, 2021.
[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *NIPS*, pp. 1-11, 2017.
[23] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," *ICLR*, 2017.