Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

DATA-DRIVEN LEARNING FOR ROBOT PHYSICAL INTELLIGENCE

by
Leidi Zhao

The physical intelligence, which emphasizes physical capabilities such as dexterous manipulation and dynamic mobility, is essential for robots to physically coexist with humans. Much research on robot physical intelligence has achieved success on hyper robot motor capabilities, but mostly through heavily case-specific engineering. Meanwhile, in terms of robot acquiring skills in a ubiquitous manner, robot learning from human demonstration (LfD) has achieved great progress, but still has limitations handling dynamic skills and compound actions. In this dissertation, a composite learning scheme which goes beyond LfD and integrates robot learning from human definition, demonstration, and evaluation is proposed. This method tackles advanced motor skills that require dynamic time-critical maneuver, complex contact control, and handling partly soft partly rigid objects. Besides, the power of crowdsourcing is brought to tackle case-specific engineering problem in the robot physical intelligence. Crowdsourcing has demonstrated great potential in recent development of artificial intelligence. Constant learning from a large group of human mentors breaks the limit of learning from one or a few mentors in individual cases, and has achieved success in image recognition, translation and many other cyber applications. A robot learning scheme that allows a robot to synthesize new physical skills using knowledge acquired from crowdsourced human mentors is proposed. The work is expected to provide a long-term and big-scale measure to produce advanced robot physical intelligence.

DATA-DRIVEN LEARNING FOR ROBOT PHYSICAL INTELLIGENCE

by
Leidi Zhao

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Engineering

Helen and John C. Hartmann Department of Electrical and Computer Engineering

August 2021

APPROVAL PAGE

DATA-DRIVEN LEARNING FOR ROBOT PHYSICAL INTELLIGENCE

Leidi Zhao

Dr. Cong Wang, Dissertation Advisor — Date
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. Edwin Hou, Committee Member — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Lu Lu, Committee Member — Date
Assistant Professor of Mechanical and Industrial Engineering, NJIT

Dr. Qing Liu, Committee Member — Date
Assistant Professor of Electrical and Computer Engineering, NJIT

Dr. Xuan Liu, Committee Member — Date
Associate Professor of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Leidi Zhao

Degree: Doctor of Philosophy

Date: August 2021

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2021
- Master of Science in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2016
- Bachelor of Science in Electrical Engineering,
  Shanghai Maritime University, Shanghai, China, 2015

Major: Computer Engineering

Presentations and Publications:

L. Zhao, C. Wang, L. Lu, "Handling Crowdsourced Data using State Space Discretization for Robot Learning and Synthesizing Physical Skill," *International Journal of Intelligent Robotics and Applications*, 4, 390–402 (2020)

J. Li, L. Lu, L. Zhao, C. Wang, J. Li , "An Integrated Approach for Robotic Sit-To-Stand Assistance: Control Framework Design and Human Intention Recognition," *Control Engineering Practice*, 107 (2021): 104680

L. Zhao, C. Wang, L. Lu, "Data-Oriented State Space Discretization for Crowdsourced Robot Learning of Physical Skills," *American Society of Mechanical Engineers (ASME) Letter in Dynamic Systems and Control*, 2021

C. Wang, B. Wang, L. Zhao, C. Maranon, N. Goswamy, J. Y. Lee, and G. Wang, "High-Fidelity Teleoperated Scaled Vehicles for Research and Development of Intelligent Transportation Technologies," *in American Society of Mechanical Engineers (ASME) Dynamic Systems and Control Conference (DSCC)*, 2020

L. Zhao, R. Lawhorn, C. Wang, L. Lu, B. Ouyang, "Synthesis of Robot Hand Skills Powered by Crowdsourced Learning," *IEEE International Conference on Mechatronics (ICM)*, 2019

L. Zhao, Y. Zhao, S. Patil, D. Davies, C. Wang, L. Lu, B. Ouyang, "Robot Composite Learning and the Nunchaku Flipping Challenge" *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3160-3165, 2018.

J. Li, L. Lu, L. Zhao, C. Wang, X. Huo, "A Human-Centered Control Framework for Robotic Sit-to-Stand Assistance" *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 845-850, 2018

L. Zhao, R. Lawhorn, S. Patil, S. Susanibar, L Lu, C Wang, B Ouyang, "Multiform Adaptive Robot Skill Learning From Humans," *American Society of Mechanical Engineers (ASME) Dynamic Systems and Control Conference(DSCC)*, 2017.

,

"Fate has no pity,
And God's night is infinite.
Your matter is time, ceaseless time.
You are each solitary moment."

–Jorge Luis Borges

ACKNOWLEDGMENT

I would like to give my heartfelt thanks to my advisor, Dr. Cong Wang, for his illuminating guidance and continuous support through each stage of the research.

I also want to express my gratitude to all my committee members: Dr. Edwin Hou, Dr. Lu Lu, Dr. Qing Liu, and Dr. Xuan Liu for the support of my research.

I appreciate the support from my beloved department and school, Helen and John C. Hartmann Department of Electrical and Computer Engineering at New Jersey Institute of Technology. Thanks for my department providing teaching assistantship to me. I also appreciate the research funding from National Science Foundation Award (Grant number 1944069). Their financial supports make my research ideas come true.

I am also extremely grateful to all the members of our CAR lab and Dr. Lu's lab who have kindly provided me assistance and inspiration during the research.

Finally, many thanks go to my family and friends for their unfailing love and unwavering support.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

For robots to become physically autonomous and replace human labors in many demanding tasks, physical intelligence is of fundamental importance. Compared to decision-making type of intelligence such as recognizing languages and determining gaming strategies, physical intelligence emphasizes physical capabilities such as dexterous manipulation and dynamic mobility. While recent research on artificial intelligence has achieved great success in decision-making tasks and shown capability of outclassing humans on gaming, diagnosis, translation, etc., the physical intelligence of robots is still at a relatively early stage. In this dissertation, we focus on data-driven learning for robot physical intelligence, especially, we want robot can learn advanced manipulation skill without case-specific engineering, and have the ability to gain ubiquitous physical intelligence.

In Chapter 1, we present a scheme of composite robot learning from human that empowers robots to acquire advanced manipulation skills that require

1. dynamic time-critical compound actions (as opposed to semi-static low-speed single-stroke actions),
2. contact-rich interaction between the robot and the manipulated objects exceeding that of firm grasping and requiring control of subtle bumping and sliding, and
3. handling of complex objects consisting of a combination of parts with different materials and rigidities (as opposed to single rigid or flexible bodies).

In Chapter 2, we present a method that allows robots to synthesize new physical skills based on knowledge learned from crowdsourced human mentors. The method features:

1. Capability of synthesizing new physical skills instead of replicating or adjusting human-demonstrated skills.
2. Sustainable management of a continuously growing massive knowledge library learned from a large number of crowdsourced human mentors.

Several simulation experiments and physical tests are conducted to validate the two proposed learning scheme. The hardware preparation, control system deployment, and environment setup are also explained in this dissertation. Chapter 3 summarizes the proposed learning schemes and gives the conclusions.

CHAPTER 2

COMPOSITE LEARNING SCHEME

2.1 Introduction

Advanced motor capabilities are necessary for ubiquitous coexistence of robots and humans. Much research on robot dynamics and control does show success in realizing hyper robot motor capabilities. Representative work includes the running and hopping humanoid robot ASIMO by Honda Motor [1], the hyper balancing legged [2][3] and wheeled [4] robots by Boston Dynamics, the high speed running cheetah robot by MIT [5], the dynamic vision guided regrasping [6], knotting [7], and pen spinning [8] robots by the University of Tokyo. Despite the application of adaptive and learning control, these works require extensive case-specific engineering that rely heavily on ad hoc models and control strategies, and lack scalability to ubiquitous applications.

Regarding the ubiquity of robot skill acquisition, a potential solution lies in robot reinforcement learning (RL) from trial and error as well as robot learning from human demonstration (LfD), which have become two hot topics in robotic research. First studied in the 1980s, LfD aims at providing intuitive programming measures for humans to pass skills to robots [9]. Among the latest and most achieved, one well-known work is presented in [10], where a PR2 robot learned rope tying and cloth folding.

However, most LfD achievements so far are for semi-static decision-making actions instead of dynamic skills, partly due to the reliance on parameter-heavy computationally-intensive (for real-time evaluation) models such as deep neural networks (DNNs). In order to make the motion presentable to an audience, typical demonstration videos feature accelerated playback rates of up to $\times 50$.

A few works in LfD do have achieved dynamic skills such as robots playing table tennis and flipping pancakes [11] as well as ball-paddling and ball-in-a-cup

tricks [12], but with recorded human demonstration as the initial trajectories for a robot reinforcement learning (RL) from (self) trial and error. The application of RL often features structurally parameterized control policies (e.g., [13][14]) in the form of the combination of a few basis elements and can thus reduce the real-time computation load. The choice of the basis elements are often quite case-specific. M.P.Deisenroth and etc. [15] gives a survey of robot RL. RL enables a robot to search for optimal control policy not from demonstration but from trial-and-error practice, with the goal of maximizing a reward function. Proper design of the reward function and the corresponding maximization strategy is another factor that is usually quite case-specific. The same authors of the survey also achieved dynamic robot motor skills such as playing table tennis [16] and throwing darts [17] via RL with motion primitives as basis elements. However, these works are mainly for stroke-based moves, and have not addressed compound actions.

Regarding these issues, we started studying a composite learning scheme, which showed success in an inverted pendulum swing-up experiment [18] and in the nunchaku flipping challenge [19].

2.2 Composite Skill Learning

So far, the majority of the robot learning from human research community has been focusing on the concept of robot learning from demonstrations (LfD). In many occasions, LfD has become a synonym of robot learning from human [9]. A few went further and explored techniques such as allowing robot learners to ask questions [20] and human mentors to give critiques [21] along with demonstration. Theories of human learning point out that effective learning needs more than observation of demonstrations [22]. In particular, explicit explanation of the underlying principals (e.g., [23]) and testing with feedbacks (e.g., [24]) are necessary in effective teaching of complex skills. Expecting a learner to master new skills solely from observing

demonstrations is analogous to learning from a silent teacher, which certainly could only achieve limited outcomes. This explains why the reinforcement learning (RL) assisted LfD shows effectiveness in learning dynamic motor skills - because the RL is in some sense a follow-up testing and feedback mechanism.

In regard to the limit of LfD, we propose a composite learning method that integrates robot learning from definition, demonstration, and evaluation.

2.2.1 Adaptive learning from definition

We use Petri nets (PN) to define compound skills that includes multiple subprocedures. The places in the PN consist of the state variables of the robot, such as posture, joint velocities, and torques. The states and transitions in a PN represent the subprocedures and the corresponding motion actions respectively. Each transition in the PN features a relatively coherent motion pattern and can be realized using a single motion/force control policy. Petri nets are abstract enough to be composed intuitively by humans, while sufficiently symbolic for machines to parse. Despite being widely used in robotics (e.g., [25][26]), Petri nets have yet not been used to teach robots dynamic skills.

Due to possible improper human description and very often the physical difference between the human mentor and the robot learner, modification of the initial definition is necessary. Starting from an initial definition provided by the human mentor, we use adaptive measures to enable autonomous correction of the initial definition. Instead of a standard 4-tuple Petri net $PN = (P, T, A, M_0)$, we introduce a 6-tuple adaptive Petri net $APN = (P, T, A, M_0, \Lambda, C)$, where P is the set of places, T is the set of transitions, A is the incident matrix that defines the relationship among places and transitions, M_0 is the initial marking, Λ is a set of firing probabilities of transitions, and C is a set of firing conditions.

An APN allows the robot learner to revise the initial definition through learning from evaluation (Subsection 2.2.3). By adjusting the P set, T set and A matrix, places

Composite Skill Learning

1 The human mentor gives initial definition of the skill using a Petri net;
2 The human mentor demonstrates the skill for multiple times and self-evaluates the demonstrations;
3 Starting from the initial definition, the robot uses the demonstration to learn control policies for each transition and the judging conditions specified in the definition;
4 The robot also learns the evaluation criteria from the mentor's self-evaluated demonstration;
5 The robot tries out the skill and uses the learned criteria to conduct self-evaluation;
6 Additional human evaluations are optional and might help improve the fidelity of the evaluation criteria learned by the robot;
7 **if** *evaluated as failure* **then**
8 The robot checks the scores of subprocedures, locates problematic spots, and modifies the initial definition by creating an adaptive Petri net;
9 Go to 5;
10 **else if** *evaluated as success* **then**
11 The robot weights up the data from the successful trials so as to improve the learned control policies and conditions;
12 After reaching a stable performance above a certain successful rate, the skill is considered learned.

and transitions can be added or dropped from the initial definition. Our previous paper [18] presented an inverted pendulum test, in which a transition is added by the learning agent to recover from a wrong state. In addition, adjustment of the firing probability set Λ and the condition set C changes the learned skill towards more suitable to the mechanical characteristics of the robot. Section 2.4 gives an example.

The state equation of the Petri net is

$$M' = M + A\mu \tag{2.1}$$

where the M is the previous marking, M' is the marking after a transition fires. μ is a column vector indicating whether the transitions fire with its boolean elements. It is controlled by the set of firing probability Λ and the set of conditions C

$$\mu = \begin{bmatrix} d_0 p_0 & d_1 p_1 & d_2 p_2 & \dots \end{bmatrix}^{\mathsf{T}} \tag{2.2}$$

where d_i is a boolean decision value indicating if the firing condition $c_i \in C$ of the ith transition is satisfied. p_i is a boolean random value that follows Bernoulli distribution $\Pr(p_i = 1) = \lambda_i$, where $\lambda_i \in \Lambda$ defines the firing probability of the ith transition.

Starting from the initial C and Λ specified by the human mentor, the robot carries out modification through trying out the skill. When a problematic transition is identified, its firing probability λ_i is updated to

$$\lambda_i^* = \kappa \lambda_i \tag{2.3}$$

where $\kappa < 1$. Once λ_i drops below a certain level, the corresponding firing condition c_i will be updated to

$$c_i^+ = \begin{bmatrix} w_1 & \cdots & w_k \end{bmatrix} \begin{bmatrix} s_i^1 & \cdots & s_i^k \end{bmatrix}^{\mathsf{T}} \tag{2.4}$$

where w_j is a weight parameter derived from the evaluation of the jth trial. s_i^j is the recorded state when firing the ith transition at jth trial. The firing probability resets when the corresponding condition is updated.

2.2.2 Learning from demonstration with data conditioning

The Petri net definition divides a compound skill in a way that each transition has a relatively coherent motion pattern and can be governed by a single control policy regressed from the human demonstration data. To avoid case-specific engineering of model-based control, we use nonparametric regression methods. Nowadays, more and more research involving nonparametric learning use deep neural networks (DNNs) with convolutional or spiking modules, taking the advantage that a large amount of training parameters (e.g., 18 million parameters in [27]) benefits the approximation of complicated state-control mappings. The price, however, is the difficulty of executing the learned control policy in real-time for dynamic actions. [28] combined DNN with parametrized policy search and obtained a model of relatively smaller scale with around 92 000 parameters. The reduced size, however, still only allows a control rate of 20 Hz, which is difficult for dynamic actions that usually require a control rate at several hundreds to over 1000 Hz.

Instead of counting on standalone LfD with a huge amount of parameters, we seek breakthrough from the power of composite learning and turn to the more computationally efficient Gaussian Process Regression (GPR), aiming at realizing a high control rate on regular control systems. GPR has a strong history in learning control of robots. One pioneering work is presented in [29]. In our work, the regression learns a mapping from the system state x to the control u. The mapping is used as a motion control policy to realize a specific transition in the PN definition. Consider a data set $\{(x_i, u_i) : i = 1, 2, \ldots, n\}$ from human demonstrations. GPR assumes that the mapping $u = u(x)$ follows a multi-variable joint distribution with certain statistical

characteristics. We apply the exponential kernel function in the state space

$$k\left(x_i, x_j|\theta\right) = \sigma^2 \exp\left(-\frac{1}{l^2}(x_i - x_j)^\mathsf{T}(x_i - x_j)\right) \tag{2.5}$$

where $\theta = \{\sigma, l\}$ includes the so-called hyperparameters to be trained, with σ being a covariance scaling factor and l being a distance scaling factor. Because the whole skill is divided into multiple subprocedures that each has a relatively simple motion pattern, there is no need to use advanced kernels (e.g., [30] Subsection 5.4.3), which lead to demanding and case-specific parameter training. The covariance matrix of the data is

$$K = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{bmatrix} \tag{2.6}$$

and the covariance matrix relating the queried state x_* to the data is

$$K_* = \begin{bmatrix} k\left(x_*, x_1\right) & \cdots & k\left(x_*, x_n\right) \end{bmatrix} \tag{2.7}$$

The control u_* for the queried state x_* can be inferred using the conditional expectation

$$\mathbb{E}\left[u_*\right] = K_* K^{-1} U_t \tag{2.8}$$

where U_t is the stack of the controls in the training data. Note that the computation load of u_* very much depends on the sizes of K and K_*, which in turn depend on the size of the data set. In order to achieve high computing efficiency for real-time control as well as improve the fidelity of the regressed mapping, we have developed a data conditioning method [31] using rank-revealing QR (RRQR) factorization. The RRQR factorization of the stack S_t of the states from the data set is in the form of

$$S_t \Pi = Q \begin{bmatrix} R_{11} & R_{12} \\ \mathbf{0} & R_{22} \end{bmatrix} \tag{2.9}$$

where Π is a permutation matrix, Q is orthogonal, and $R_{11} \in \mathbb{R}^{m \times m}$ is well conditioned. The columns in S_t identified by the first m columns of Π form a well conditioned subset. Various algorithms are available to compute an RRQR factorization, providing different lower bounds of R_{11}'s smallest singular value [32]. The subset selected features improved condition number and leads to more reliable regression fidelity, while takes only a fraction of the original computation.

2.2.3 Learning from evaluation

After acquiring the skill definition and regressing control policies from human demonstration, we use evaluations to tune the learned skill. The human mentor and the robot learner often have nontrivial physical difference, and the skills learned right off the human demonstration are often not optimal or even less feasible to the robot learner. Learning from evaluation handles this problem. When the robot tries to carry out a learned skill, both the success of the whole skill and the performance of each subprocedure are evaluated. In order to avoid case-specific engineering, the scoring formulae are not explicitly specified by the human mentor. Instead, the human mentor labels his/her demonstration with success/failure flags and performance scores. The learning agent learns the scoring criteria from the labeled data and use them to self-evaluate the robot's practices, which always have variations due to the dynamic and compound nature of the skills. The evaluation result is in turn used to refine the learned skills by taking in the data from more successful practices, while the data from lower scored demonstrations are less weighted. Examples are discussed at the end of Section 2.4.

2.3 Inverted Pendulum Swing-up and Balancing

Due to its dynamic nature, the swing-up and balancing of an inverted pendulum has been a popular test for robot learning strategies. As shown in Figure 2.2, a

6-axis AUBO i5 robot manipulator is used to carry an inverted pendulum. The robot provides an open-architecture control interface driven by a Controller Area Network (CAN) bus, allowing torque, velocity, or position level control. A National Instruments CAN PCI interface and MATLAB/Simulink Vehicle Network Toolbox are used to facilitate a CAN-based real-time control system at a sampling rate of 1 kHz.

Figure 2.1 shows the Petri net definition of the skill, which consists of the swing-up and balancing phases. P_0 denotes the initial state, P_1 and P_2 represent the swung-up and balanced states (with success/failure judging conditions) respectively. $P_{F(\text{success})}$ and $P_{F(\text{failure})}$ are the final success or failure status. The state variables in the places are the angle θ of the pendulum and its angular velocity ω. t_0 denotes the starting move. t_1 is the switching action from swing-up to balancing. t_2 and t_3 are the repeating swing-up and balancing moves respectively. t_4 or t_5 fires when the swing-up state or the balanced conditions cannot be reached over time and lead to the $P_{F(\text{failure})}$ state. The successful stop action t_6 fires when the balanced state is maintained steadily for a certain amount of time. Transitions t_4, t_5 and t_6 all lead to the end of the maneuver.

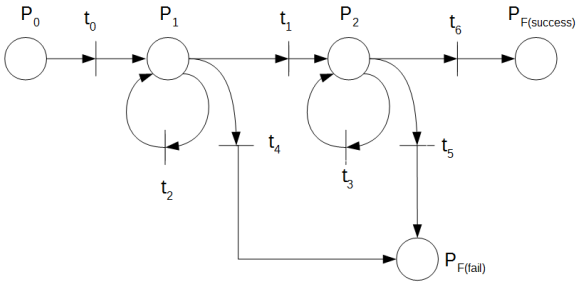

Figure 2.1 The skill of swing-up and balancing of an inverted pendulum defined by a human mentor.

Multiple demonstrations of the skill are performed by a human mentor using joystick control. The human mentor labels the individual performances in his/her own demonstrations as success or failure. The control laws of each transition specified in the Petri net are trained using data from the successful demonstrations. In addition,

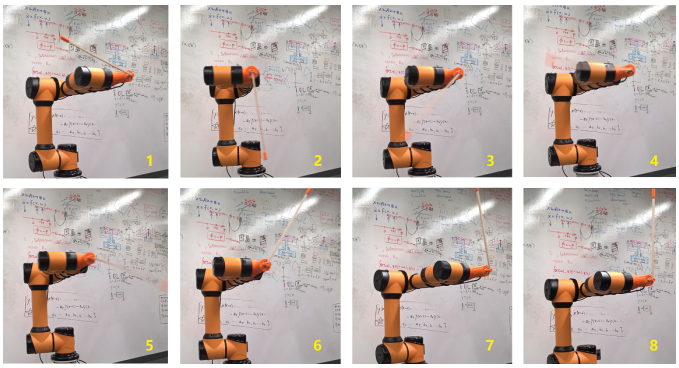

Figure 2.2 A robot learned to swing up and balance an inverted pendulum.

the learning agent learns the judging criteria from the labeled data and allows the robot to grade its own performance. Starting from the initial Petri net definition and demonstration data, the robot attempts repeated trials of the skill. Each time a trial is completed, the robot grades its own performance using the learned criteria.

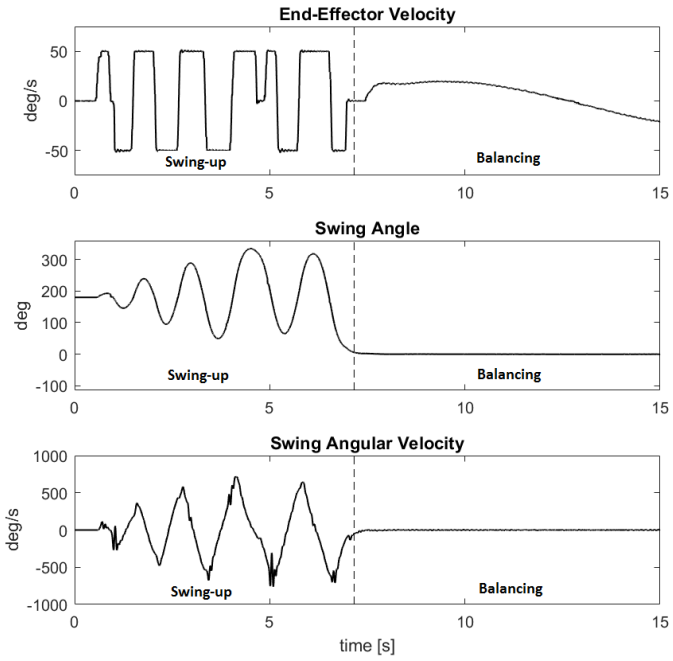

Figure 2.3 Motion variables of the robot and the pendulum.

Depending on the trial results, the robot may modify the initial Petri net definition (Figure 2.1) provided by the human mentor using the APN method introduced earlier. As an example, in the case that the human mentor defined

the condition of firing t_1 (swtiching to balancing) to require exact upright position and absolute zero angle velocity of the pendulum (i.e., a perfect swing-up), the system would almost never get the chance to proceed to P_2, and the process would end up failing (after repeating t_2 a certain number of times) in every trial. Upon the detection of constant failure, the robot starts to modify the original definition using the proposed APN. First, the learning agent identifies that transition t_1 has never been fired, the condition to fire it has never been satisfied, and place P_2 has never been visited. As t_1 is already defined as the only transition connecting P_1 and P_2 (as specified in A), no new transition is added, and adjustment of Λ would induce no effects. That leaves the learning agent to adjust C by examining the data collected from human demonstration. In particular, an SVM is used to learn a new condition (specified by the state variables) by regression from the demonstrated switching action corresponding to t_1. The resulting APN features a feasible definition of C and allows successful triggering of t_1. Figure 2.2 shows a successful performance carried out by the robot after the learning. Figure 2.3 shows the motion variables of the robot and the pendulum, in which the transition from swing-up to balancing can be clearly seen.

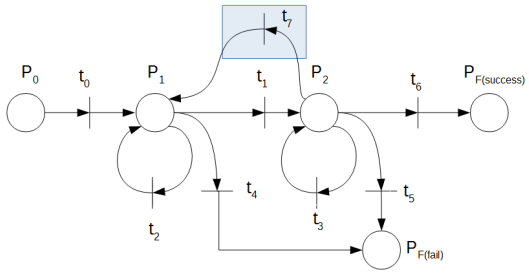

Figure 2.4 Revised Petri net.

Sometimes the process is also affected by excessive external disturbance and forced out of a known sub-procedure. In particular, before t_6 is fired (upon steady balancing over a certain amount of time), the balanced status can be broken due to external disturbance, which might be too intense and the status cannot be recovered by firing t_3. Such a situation is not specified in the initial definition. In order to recover,

the learning agent first identifies P_2 in the Petri net as where the issue occurs. The state variables after the incident occurs are examined. Specifically, their correlation to each place in the Petri net is evaluated, which indicates that the status belongs to P_1. The learning agent then adds a new transition t_7 from P_2 to P_1 as shown in Figure 2.4, which recovers the process by backing up to the swing-up phase.

2.4 The Nunchaku Flipping Challenge

We hope the composite learning scheme empowers robots to acquire advanced manipulation skills that require

1. dynamic time-critical compound actions (as opposed to semi-static low-speed single-stroke actions),
2. contact-rich interaction between the robot and the manipulated objects exceeding that of firm grasping and requiring control of subtle bumping and sliding, and
3. handling of complex objects consisting of a combination of parts with different materials and rigidities (as opposed to single rigid or flexible bodies).

So we also introduce the "nunchaku flipping challenge", an extreme test that includes hard requirements on all three elements listed above.

2.4.1 Hardware preparation

A robot arm and real-time control system We use a 6-joint AUBO i5 robot arm. It features a modular design similar to the popular UR series by Universal Robots, while provides a much simpler open control interface that allows end-users to fully access real-time position/velocity/torque control. The control deployment (Figure 2.5) is based on a target computer directly connected to the robot arm via a Controller Area Network (CAN bus) cable. Other than a National Instruments PCI-CAN interface, no additional interfacing hardware is used. MATLAB/Simulink is used to implement sensing, control, and safety algorithms. The MATLAB Vehicle Network Toolbox is used to facilitate the CAN communication protocol. The sampling rate of the control system is 1 kHz.

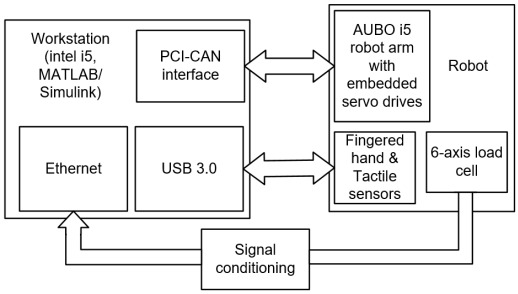

Figure 2.5 Control deployment of the test setup.

A bionic robot hand In order to facilitate advanced manipulation skills involving finger actions, we developed a bionic robot hand with haptic sensors. The hand features a bionic five-finger design and tendon-driven actuation. The majority of the hand is 3D-printed, including the palm and finger segments in PET, finger joints in the rubbery TPU (for auto extension), and a motor pack in stainless steel. TakkTile sensors developed by Righthand Robotics are used as haptic sensors. They are built up on the NXP MPL115A2 MEMS barometer by casting a rubber interface [33]. In addition, an ATI 6-axis load cell is installed at the wrist to perceive the centrifugal force brought by the motion of any payload manipulated by the hand.

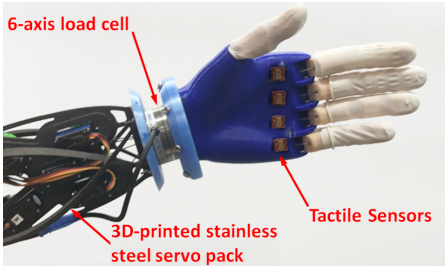

Figure 2.6 A fingered robot hand with tactile sensors.

Motion capture systems Motion capture is used in learning from demonstration. Accurately capturing a highly dynamic skill is a challenge. It is also important to provide an intuitive interface for efficient teaching. We have experimented several options.

The Microsoft Kinect seems to be a first choice. It has been recognized as a top product among commercial camera and image processing-based systems. Not requiring markers or hand-held gadgets makes it very intuitive for the mentor to demonstrate a skill. However, such camera-based systems suffer from limited sampling rate (usually up to 30 frames per second) and considerable delay caused by image processing (usually takes up to an entire sampling period), plus low accuracy as reported in [34], which in turn make the velocity estimation difficult. We tried to compensate these problems, otherwise known as visual sensing dynamics using a predictive filtering technique [35]. First, the position signal $s(t)$ being sensed is decomposed using it's Taylor expansion with respect to time

$$s\left(t_0+t\right)=\sum_{c=0}^{r} s^{(c)}\left(t_0\right)\frac{t^c}{c!}+s^{(r+1)}\left(t_0+\xi\right)\frac{t^{r+1}}{(r+1)!} \tag{2.10}$$

where r is the order of the expansion, and $\xi \in (0,t)$. The expansion can be written into a state space model:

$$x\left(i+1\right)=\begin{bmatrix} 1 & T & \cdots & \frac{T^r}{r!} \\ & 1 & \cdots & \frac{T^{r-1}}{(r-1)!} \\ & & \ddots & \vdots \\ & & & 1 \end{bmatrix} x\left(i\right)+\begin{bmatrix} \frac{T^{r+1}}{(r+1)!} \\ \frac{T^r}{r!} \\ \vdots \\ T \end{bmatrix} u\left(i\right) \tag{2.11}$$

with an output

$$y\left(j\right)=\begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} x\left(j-L\right)+v\left(j\right) \tag{2.12}$$

where the state vector $x=\begin{bmatrix} s & s' & \cdots & s^{(r)} \end{bmatrix}^{\mathsf{T}}$ contains s and its derivatives. i is the time step index. T is the algorithm sampling time, which is much shorter than the camera sampling time. u comes from the residual term in Equation (2.10). It is treated as an unknown input, and handled using an *equivalent noise approach* [36]. y is the position identified by image processing. v is the artifacts and rounding noise.

$j = N, 2N, 3N, \ldots$ is the index of the camera sampling actions. L is the delay caused by image processing.

A dual-rate adaptive Kalman filter is then applied to Equations (2.11) and (2.12) to compensate for the delay and recover the information between sampling actions. Despite reported success of this type of compensation techniques [35], we found that it still requires the camera to sample at least 15 times faster than the desired bandwidth of the motions being sensed. For the highly dynamic maneuvers targeted in our work, such a limit excludes the use of any commercial camera and image processing-based motion capture systems.

Another type of non-contact motion capture systems is the ones using active infrared makers and infrared sensors. Consumer level products of such type include the Nintendo Wii and Sony PS Move, which unfortunately are of very limited accuracy [37]. Meanwhile, the high-end options such as the one introduced in [38] and its commercial peers (e.g., PhaseSpace and NDI Optotrak), although are capable of obtaining very high quality measurement, are much beyond the space and cost considerations in our long term goal of making the technology available to everyday life.

A balanced choice between cost and capability is a mechanical motion capture system in the form of a passive multi-bar mechanism equipped with rotation sensors at the joints. Compared to the previous two options, such systems, either the commercial ones such as Geomagic Touch or a customized design (Figure 2.7 left) provide both an feasible cost and sufficient capability to our application. Although the usage is not as intuitive as non-contact motion capture systems, the additional difficulty is acceptable. In addition, a sensing glove with Flex sensors is used to capture the motion of the fingers. The glove is also equipped with vibrating motors to provide tactile feedback to the user (Figure 2.7 right).

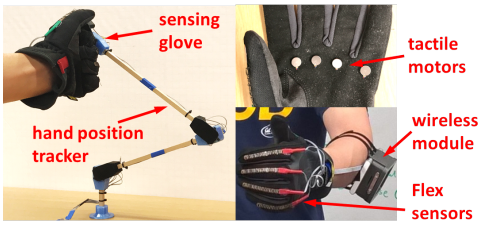

Figure 2.7 Motion capture gadgets used in the tests.

2.4.2 The experiment

Nunchaku is a traditional Okinawan martial arts weapon widely known due to its depiction in film and pop culture. It consists of two rigid sticks connected by a chain or rope. Among the numerous tricks of using nunchaku, the flipping trick as shown in Figure 2.8 is one that puts hard challenges to all three elements we consider in advanced manipulation skills, i.e., dynamic maneuver, hand-object contact control, and handling partly soft partly rigid objects. The trick includes three subprocedures: swing-up (1–3 in Figure 2.8), chain rolling (4–6), and regrasping (7, 8).

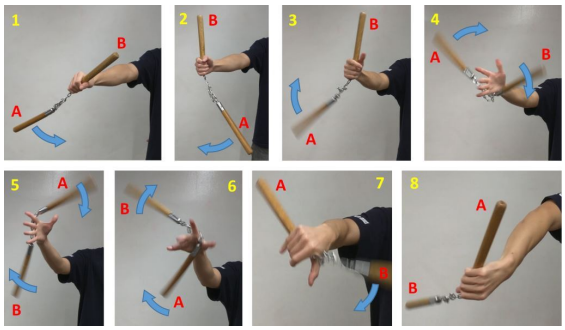

Figure 2.8 The nunchaku flipping trick.

With the composite learning scheme, the nunchaku flipping trick is first described by a human mentor using an initial Petri net definition. As shown in Figure 2.9, P_0 is the initial state, in which the robot hand holds one of the sticks and is in no motion. When the start-of-motion transition t_0 fires, the robot begins the swing-up procedure. The swinging t_1, the stop motion t_6, and the hand-releasing action t_2 fire based on the probability and the judging conditions in P_1. If the running time goes beyond

a threshold, the action stops by firing the stop motion t_6. If the sensor reading is below a certain level, the swing t_1 fires and the amplitude of the swing is increased. After the sensor reading exceeds the threshold, t_2 has a possibility to fire. Similar to the swing-up procedure, when the hand-releasing action t_2 successfully fires, the robot goes on to chain-rolling. The back palm contact control t_3, the stop motion t_7, and the regrasping action t_4 fire based on the probability and the judging conditions in P_2. The robot regrasps by firing t_4. If the regrasping is successful according to the condition in P_3, stop motion t_5 fires and leads to the final success $P_{F(\text{success})}$. Otherwise, stop motion t_8 fires and leads to the final failure $P_{F(\text{fail})}$. The possibilities and conditions could change during the learning process.

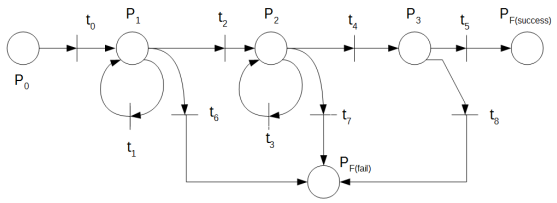

Figure 2.9 Initial Petri net definition of the nunchaku flipping trick.

The bionic robot hand described in Subsection 2.4.1 is installed on the robot arm to resemble human hand maneuvering. In order to keep a reasonable fidelity to human sensory control, no sensor is installed on the nunchaku. The motion of the sticks and the chain is perceived by the haptic sensors and 6-axis load cell in the robot hand. In addition, without explicit inference of the position, attitude, and layout of the sticks and the chain, the sensor readings are directly mapped to the motor controls of the fingers and the arm joints by the learning algorithm. Such an end-to-end learning scheme has earned increasing preference recently and is believed to be a good approximation of human neural response (e.g., [28]).

Multiple demonstrations of the nunchaku flipping trick are performed by a human mentor and recorded by the motion capture systems. The human mentor labels if each demonstration is a success and scores the performance. The control policies of

the transitions and the judging conditions are learned from successful demonstrations weighted by their scores. The grading criteria for robot self-evaluation are learned from both successful and failed demonstrations. Starting with the initial definition, the robot conducts multiple trials. After each trial, the robot grades its own performance using the learned criteria. The final score and the score of every transition are given to determine if the trial is a success and which part in the definition should be adjusted.

Such adjustment is important because of the physical differences between the human mentor and the robot learner. This is especially true for the ending part of the swing-up which requires certain vertical speed to enter chain-rolling. The human mentor tends to use a sudden jerk-up to realize this part. Despite being a small move, this action is at the border of the robot's mechanical limit. As a result, the learning agent avoids learning from demonstrations featuring this move because of a low success rate during the trial runs, while weights up the data form demonstrations with a more back-and-forth type swing-up, which achieves much more successes in the trial run evaluations. Similar situation applies to the condition of switching from swing-up to chain rolling. The initial switching condition learned from human demonstrations is not the optimal for the robot, which can be adjusted through learning from evaluation during trial runs.

2.5 Conclusions

This chapter introduces a composite robot learning scheme which integrates adaptive learning from definition, nonparametric learning from demonstration with data conditioning, and learning from evaluation. The method tackles advanced motor skills that require dynamic time-critical maneuver, complex contact control, and handling partly soft partly rigid objects. We use classic inverted pendulum test to validate the proposed learning scheme. In addition, we also introduce the "nunchaku flipping challenge", an extreme test that puts hard requirements to all these three

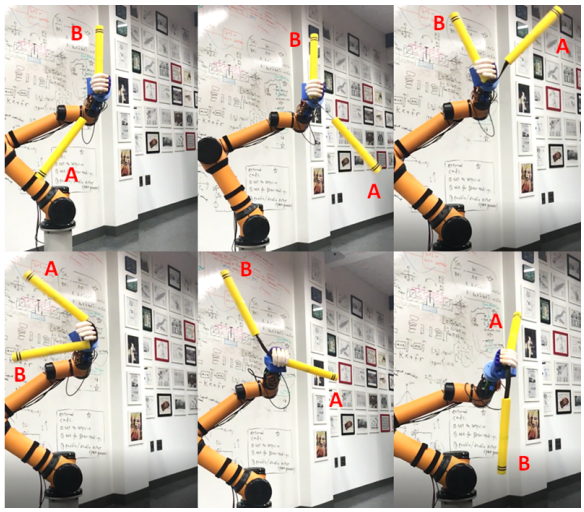

Figure 2.10 Nunchaku flipping learned by a robot.

aspects. Details of the hardware preparation and control system deployment of a physical test are explained. The proposed robot learning scheme shows promising performance in the challenge.

CHAPTER 3

ROBOT PHYSICAL INTELLIGENCE POWERED BY CROWDSOURCED LEARNING

3.1 Introduction

Physical intelligence is of fundamental importance for robots to interact with the real world. Compared to cognitive intelligence for tasks such as language processing and image recognition, robot physical intelligence for tasks such as dexterous manipulation and dynamic mobility is still at an early stage. Methods for enabling robot physical intelligence generally sit in two categories. The first category is motion planning based on analytical physics models. Representative work includes the running MIT cheetah robot [5], the dynamic vision guided baseball [39], and so on. Despite the impressive motion capabilities, such methods require extensive case-specific engineering that relies heavily on ad hoc and complex models, which limit ubiquity.

Meanwhile, learning-based methods, such as learning from demonstration [40] and reinforcement learning [41] allow robots to acquire physical skills through mimicking a mentor or self-directed trying. Robot learning based on data-driven methods greatly reduces the reliance on analytical models derived from laws of physics. Nevertheless, certain limitations exist. Most state-of-the-art methods for robot learning physical skills follow a "policy search" framework, which instead of recording and replaying demonstrated/self-explored successful motion, aims at producing a control policy that can handle the variations when fulfilling the skills. Many of these control policies are in the form of parameter-heavy structures such as a deep neural network (DNN) (e.g., [10]). Such control policies require lengthy training of a massive amount of parameters, which hampers them from handling dynamic manipulation and realizing online real-time feedback control. The latter is especially necessary to open-loop unstable tasks such as in-hand manipulation. Another popular practice is to formulate

control policies as combinations of a few base elements such as the "motion primitives" (e.g., [13, 14]). Training such control policies requires significantly less computation, but the choice of the base elements and the design of the reward function require very task-specific engineering and jeopardize ubiquity. Recently, deep reinforcement learning has gained great attention. It provides better ubiquity by approximating the reward function using a DNN (e.g., [42, 43]). Limitation of such approaches is brought by the lengthy training curves required by every new task. A representative example is the recent work by OpenAI on using a Shadow robot hand to learn in-hand rotation of a cube. Both the heavy training process and the need of task-specific engineering have been marked by experts [44].

In addition to ubiquity and computation, another consideration is over the data source of skill acquisition. For complex physical skills, especially those involving dynamic maneuvers, it is impractical to rely solely on reward-driven self-practice (i.e., basic reinforcement learning) without advisory from mentors [22]. Over the past decade, large-scale datasets obtained from a group of mentors (crowdsourced) have greatly accelerated the advancement of artificial intelligence [45]. Together with crowdsourcing, human computation [46] has become an increasingly popular way to provide mentorship to learning agents. The concept is particularly popular in the field of cognitive machine intelligence such as language processing and image recognition. A classic example is Google Images, whose AI system is trained using a massive amount of sample images reviewed and labeled by human participants from over the world. Other successes include translation, merchandise review, and medical diagnosis [47, 48, 49, 50].

Despite its promising potentials, replicating the success of crowdsourced human computation in robot physical intelligence has not been explored much. Challenge first comes from building intuitive control interfaces. Pioneering work regarding this topic includes MIT's Homunculus project [51] that develops a virtual reality remote

robot control environment and Stanford's Roboturk project [52] that uses smartphones as remote robot controllers. Another challenge is on sustainable management and efficient utilization of the collected large datasets. Published projects that apply crowdsourced human computation to generate robot physical skills include grasping novel objects [53, 54] and assembly [55]. So far, these projects are limited to realizing semi-static operations by searching for similar or available solutions in the dataset.

Our research on crowdsourced robot learning of physical skills develops a unique learning scheme that brings contributions with the following capabilities:

1. Allowing robots to learn physical skills from a group of human mentors constantly through demonstrations.
2. Sustainably managing a large amount of data that is constantly collected from the crowdsourced mentors.
3. Synthesizing dynamic physical skills that have been never or only partly demonstrated, without the need of heavy training/re-training.
4. Efficient handling of high-dimensional large datasets.

3.2 A Crowdsourced Learning Scheme

We formulate a physical robot skill as a control action that drives a robot and an object being manipulated from a given initial state to a desired final state. By including state variables such as velocity and force, dynamic skills that require time-critical maneuvers can also be synthesized. Every human demonstration is represented as a $\{u(t), x(t)\}$ pair of time sequences, where $u(t)$ is the control action (e.g., robot joint torque, velocity command) and $x(t)$ is the state response (e.g., the position and orientation of the object being manipulated). In the state space, a state response starts from an initial state, goes through a series of via states and arrives at a final state. Demonstrations covering various areas and transitions in the state space form **a map similar to a bus and train map**. Synthesizing a skill becomes similar to finding a travel plan of buses and trains for going from a starting location to a destination. New

skills can be created by cutting and stitching (with additional editing) the control actions of the demonstrated trajectories.

Three issues need to be addressed first to apply the proposed idea:

1. As new demonstrations are constantly collected from the crowdsourced mentors, the knowledge library cannot accommodate unlimited number of trajectories.
2. The actual initial state and the desired final state usually do not sit exactly on any demonstration trajectory.
3. The demonstration trajectories usually do not have exact intersecting points in the state space (analog of the transfer stations on a bus and train map).

In regard to these issues, we introduce a *two-step skill synthesis* method using state space discretization, which discretizes the state space into a finite number of cells. Figure 3.1 shows the functional block diagram of the method. In the first step, two states siting in the same cell are considered to be at the same location in the state space. The demonstrations archived in the library are registered to a graph representation (Figure 3.2). The nodes in the graph represent the cells in the state space, with each archived demonstration corresponds to a series of edges connecting multiple nodes. Maneuver segments from new demonstrations connecting the same cells are compared, with the optimal one registered to the edge. While new demonstrations can be collected unlimitedly, the maximum number of maneuver segments archived in the knowledge library is fixed to $n \times (n-1)$, where n is the total number of cells and can be as large as the available computation resource allows. In this way, the maximum load of data handling for skill synthesis is fixed.

The first step of skill synthesis produces a "rough" skill in the form of a series of edges in the graph representation, starting from the node containing the initial state, ending at the node containing the final state, and going through a series of via nodes. In the second step, statistical nonparametric inference is applied to the demonstration data registered to the edges and produces the control action that drives the robot and the object being manipulated from the exact initial state to the exact final state through a series of via states that belong to the via nodes.

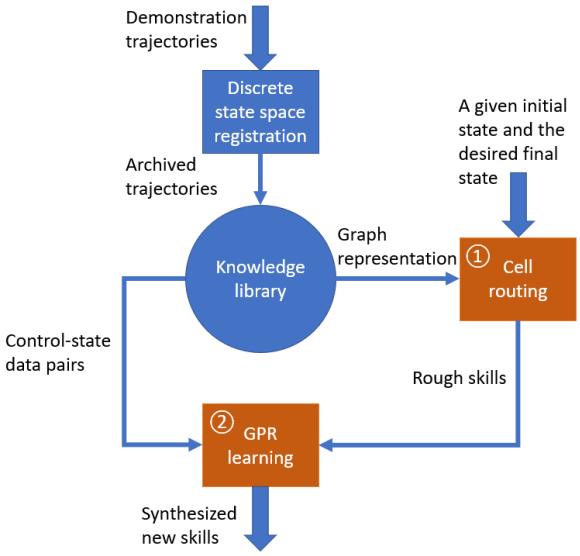

Figure 3.1 Two-step skill synthesis.

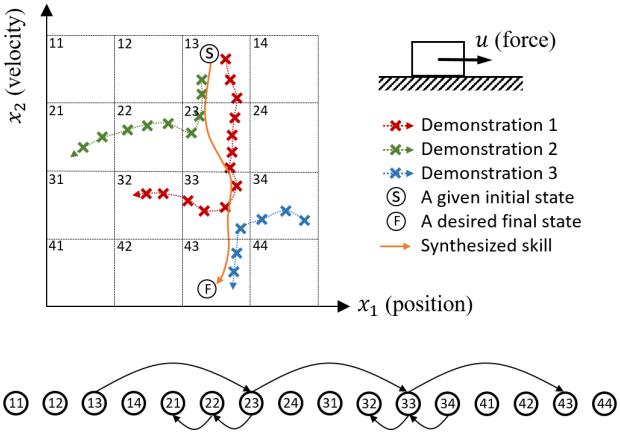

Figure 3.2 An example of state space discretization and its graph representation.

3.2.1 An elementary example

To better explain the proposed method, consider an example with a simplified configuration as shown in Figure 3.2. A two-dimensional state space is discretized to 16 cells. The two state variables, x_1 and x_2 could be the position and velocity of a linearly moving object being pulled and pushed by a force (applied by a robot). Three demonstration trajectories are collected in the library and registered to seven edges in the graph representation. Note that: a) the edge connecting nodes ⑬ and ㉓ comes from two demonstrations; b) the demonstration trajectories do not exactly intersect in the state space but are joined in the graph representation; and c) the initial/final states do not sit on any demonstration trajectory in the state space but do belong to edge-connected nodes in the graph representation. A rough skill is first found as a path in the graph representation: ⑬ → ㉓ → ㉝ → ㊸. Then, the control action $u(t)$ is created by applying statistical inference (Subsection 3.2.4) to the demonstration data pairs $\{u, x\}$ registered to edge ⑬ → ㉓, ㉓ → ㉝, and ㉝ → ㊸.

In order to extend this elementary example to a practical full solution, several issues need to be further addressed:

1. The discretization method in the example (a full factorial design) becomes extremely inefficient when the dimension of the state space increases. We introduce an efficient discretization scheme using low-discrepancy pseudo-random sequences (Subsection 3.2.2).
2. A planning method is needed to produce the rough skill from the graph representation of the knowledge library (Subsection 3.2.3).
3. Statistical inference is needed to produce the final control action (Subsection 3.2.4).
4. New demonstrations need proper processing and registration to the knowledge library (Subsection 3.2.5).

3.2.2 State space discretization using pseudo-random sequences

In the proposed robot learning scheme, physical skills are defined using state transitions in the state space. Some state space discretization techniques actually exist for

reinforcement learning (RL) [56, 57]. RL applications usually tackle a single control task and these techniques discretize the state space according to the (one) control policy for the task. Meanwhile, our proposed learning scheme aims at treating various control tasks ubiquitously, which allows different control actions to correspond to the same state. In terms of synthesizing skills of a greater variety, it is desirable to have data populate the state space thoroughly, which in turn requires a uniform and thorough discretization of the state space. In addition, for the sake of computational affordability, the state space should be discretized with as few cells as possible.

The total number of cells of a full factorial design (as used in the elementary example earlier) increases exponentially as the dimension of state space grows, which leads to excessive load of data handling. In addition, although a full factorial design provides a seemingly good uniformity, the coverage is actually quite inefficient. As illustrated in Figure 3.3 (a), consider a three-dimensional state space discretized by a four-level full factorial design. The total number of cells is $4^3 = 64$. However, the cells' projection to a lower-dimensional subspace suffers severe overlapping. In this case, a two-dimensional subspace has only $4^2 = 16$ cells and a one-dimensional subspace has only 4 cells.

An intuitive better choice is a random discretization of a uniform distribution. The randomness minimizes the overlapping in the cells's projection to lower-dimensional subspaces. However, governed by the law of large number for random variables, a good uniformity can hardly appear if only a limited numbers of cells can be computationally afforded (Figure 3.3 (b)). Inspired by [58], a low-discrepancy pseudo-random sequence is used to discretize the state space. Compared to the full factorial design, low-discrepancy pseudo-random sequences ensure the uniformity through all lower-dimensional subspaces. Low-discrepancy pseudo-random sequences provide the same excellent uniformity of random distributions, but are actually deterministic instead of random. Even when the available computing

resource only allows a small amount of cells, a discretization created using a low-discrepancy pseudo-random sequence still provides high uniformity over subspaces of lower-dimensions (Figure 3.3 (c)). In other words, it gives the most uniform discretization that can be achieved with a certain computing resource.

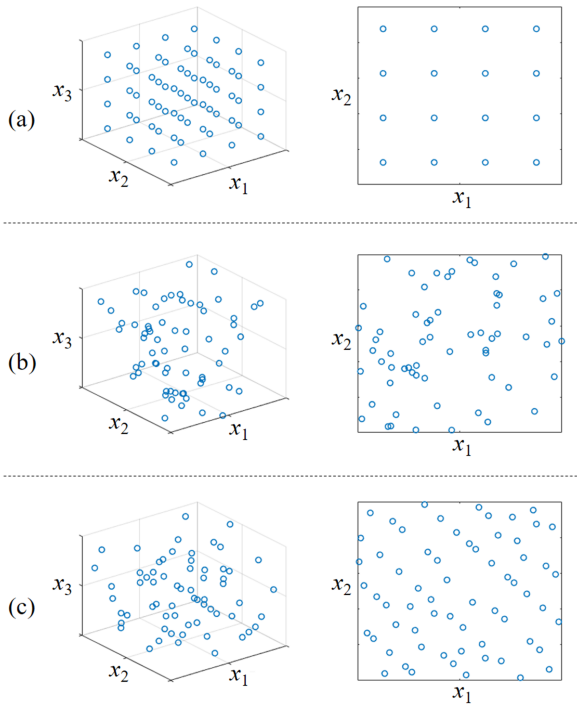

Figure 3.3 3D state space discretization using 64 points of different distributions and their projections to a 2D subspace - (a) full-factorial design, (b) uniform random numbers, (c) low-discrepancy pseudo-random sequence.

In our work, the Sobol sequence is adopted to discretize the state space. It is a widely used low-discrepancy pseudo-random sequence [59]. The sequence is generated using the so-called direction numbers $v_{i,j} = m_{i,j}/2^i$, where $m_{i,j}$ is an odd integer from a specific recurrence sequence. The component of the k'th point in the Sobol sequence on the j'th dimension is given as $s_j^{(k)} = b_1 v_{1,j} \oplus b_2 v_{2,j} \oplus b_3 v_{3,j} \oplus \cdots$, where b_i is the i'th bit of k's binary code, and $\oplus$ is the bit-by-bit exclusive-or operator. More details can be found in [60].

Despite the efficient coverage, use of a pseudo-random sequence to discretize the state space brings difficulty to registering a state to a cell, which is in essence a nearest

neighbors problem. In the case of a full factorial design, thanks to the well-aligned and naturally sorted distribution, it is easy to locate the cell that a state belongs to. Meanwhile, the discretization created by a pseudo-random sequence is not aligned. The order of the cells, though actually deterministic, follows a sophisticated pattern, and brings a challenge.

To tackle this problem, we make use of the property of a Sobol sequence - that its uniformity is well preserved in its projection to any lower-dimensional subspace. As an explanatory example, consider a two-dimensional state space discretized using a two-dimensional Sobol sequence of 100 points ranging from $[0, 0]$ to $[1, 1]$, and one would like to locate the cell that the state $x = [0.3, 0.2]$ sits in. Thanks to the uniformity property of Sobol sequences, if the Sobol sequence is sorted according to the first state variable, the state should sit closest to the points around the $100 \times 0.3 = 30$th position in the sorted sequence. Then, the cell that the state sits in can be located quickly by searching in a small range nearby (e.g., the 27th to 33rd points in the sorted Sobol sequence). An example of a trajectory registered to a discrete state space formed using a Sobol sequence is shown in Figure 3.4. The pseudo-code below Figure 3.4 explains the implementation.

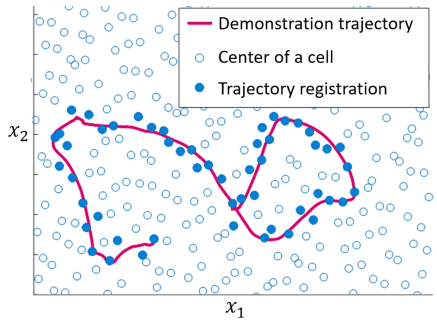

Figure 3.4 A trajectory registered to a discrete state space.

Registering a state $x = [x_1, x_2, \cdots, x_N]$ to a state space discretized by a Sobol sequence

1 Discretize the N-dimensional state space using an N-dimensional Sobol sequence of n points $\{s^{(1)} \quad s^{(2)} \quad \cdots \quad s^{(n)}\}$.

2 Sort the Sobol sequence in ascending order according to the value of the first state variable $s_1^{(k)}$.

3 Calculate the search center as $x_1 \times n$.

4 Specify an error boundary $e = [e_1, e_2, \cdots, e_N]$. The search interval is $[(x_1 - e_1) \times n, \quad (x_1 + e_1) \times n]$.

5 **for** $i = (x_1 - e_1) \times n$ *to* $(x_1 + e_1) \times n$ **do**

6 Check if $\left|s_j^{(i)} - x_j\right| < e_j$ for $j = 2 \sim N$;

7 **if** *only one* $s^{(i)}$ *satisfies the condition* **then**

8 Register x to the point.

9 **if** *more than one* $s^{(i)}$ *satisfy the condition* **then**

10 Examine the Euclidean distance between the x and the candidate $s^{(i)}$'s;

11 Register x to the point of the shortest distance.

3.2.3 Synthesis of the rough skill

With the discretized state space, the knowledge library can be represented as a directed graph. Rough skills can be synthesized in the form of paths through the nodes of the graph. This transforms (the first step of) skill synthesis to a classic single-source shortest-path problem. With every node in the graph representing a small region (cell) in the state space, an edge connecting two nodes indicates that certain knowledge of control is available to drive the system from a state in the first cell to a state in the second cell. A weight can be setup for each edge to mark the merit (e.g., time, mechanical/electrical load, etc.) that the maneuver takes.

In our work, we adopt the well-proven Dijkstra's algorithm [61] to find a path in the graph. Dijkstra's algorithm describes a graph using the adjacent matrix, whose (i, j) element is the weight of the edge connecting node i and j. In our application, there can be segments from multiple demonstration trajectories connecting the same cells. The segment carrying the best merit is archived to the knowledge library and registered in the adjacent matrix.

3.2.4 Generation of the control action

With the rough skill in the form of a series of edges in the graph representation of the knowledge library, step two of skill synthesis applies statistical inference to produce the control action. Gaussian Process Regression (GPR) is used. As a nonparametric learning method (as opposed to regression of structured models), GPR has shown great potential in robot learning control [29][62]. In our work, GPR is used to learn a control law $u(x)$ in the form of a mapping from the state x of the robot and/or the object being manipulated to the control action u of the robot. The $\{x, u\}$ pairs from the archived demonstration trajectories are used as the training data. The entire control action of a synthesized skill can be considered as a series of GPR mappings trained using data from the selected segments of the demonstration trajectories. During the execution of the synthesized skill, the control switches from one GPR mapping to another sequentially.

In order to realize a sustainable management of the knowledge library, instead of training one GPR mapping for an entire demonstration trajectory, each GPR mapping is trained only using the data from a trajectory segment connecting two cells in the state space. In this way, while new demonstration trajectories can be collected unlimitedly, the maximum number of GPR mappings is fixed to $n \times (n-1)$ where n is the total number of cells in the discretized state space. Such a strategy is particularly helpful to crowdsourced learning, whose power comes from a massive and constantly

growing amount of demonstrations. It also reduces the computation load of running a GPR mapping in real-time. Without the need of discussing many details of the GPR algorithm, a major step in evaluating a GPR mapping requires evaluating and inverting an $m \times m$ data covariance matrix, where m is the number of training data points. Consider a demonstration trajectory of 1000 data points, a GPR mapping trained using all of its data would require the evaluation and inversion of a matrix of the size 1000×1000. Assume that a trajectory has ten 100-point segments each connecting two cells in the discretized state space. A GPR mapping trained from one of the segments only requires the evaluation and inversion of a 100×100 matrix. In addition, the computation and memory resource required to handle all ten mappings (assuming that all segments are archived) is only a fraction of that required to handle a GPR mapping trained from the entire trajectory.

3.2.5 Processing new demonstration trajectories

One goal of processing new demonstration trajectories is to control the volume of the knowledge library so that its size does not grow unlimitedly as new demonstrations constantly being collected. When a new demonstration trajectory is collected, its data points are first indexed to the cells of the discretized state space. Then, every trajectory segment that connects two cells is examined by a merit criterion, such as the time consumption, mechanical and electrical load to the robot. If the segment is better than the one previously archived, it replaces the latter. Otherwise, it is discarded. Such a scheme avoids comparing a new demonstration to all previously collected demonstrations and limits the total number of trajectory segments archived in the library to $n \times (n-1)$, where n is the total number of cells of the discretized state space and can be as large as the available computation resource allows.

3.3 Validation I - The Fidgeting Test

We validate the proposed method using a test of robot in-hand manipulation. Compared to grasping and picking, in-hand manipulation requires more advanced hand skills [63]. Lack of such skills has been a bottleneck hampering the use of robots to automate many operations. An example is the Amazon warehouses. Despite the high level of warehouse automation, the packaging sections remain manual with extreme workload [64], partly due to the requirement of advanced hand skills for handling a large variety of objects. Some most achieved research on learning-based robot in-hand manipulation includes the use of reinforcement learning to realize robotic in-hand rolling [65], which requires a trial-and-error improvement process. We believe that crowdsourced learning provides a chance to realize generic robot in-hand manipulation of arbitrary objects.

A virtual reality environment A virtual reality (VR) environment is developed using Unity to facilitate the validation. Over recent years, consumer VR electronics and development software have advanced quickly and can now provide high quality motion sensing and real-time human-machine interaction. Compared to other human demonstration interfaces, such as lead-through teaching, VR-based interfaces have earned increasing favor in robot teleoperation [51] and robot learning from humans [66].

In our work, a simulated three-finger robot hand is built in the VR environment to manipulate an object. The robot hand runs in two modes:

1. Teleoperated mode, in which the robot hand is controlled by a human mentor. The teleoperated motion and the response of the object being manipulated are recorded as demonstration data.
2. Autonomous mode, in which the robot hand executes a synthesized skill autonomously.

HTC Vive headsets are used by human mentors to visually access the simulation. A Leap Motion sensor is mounted on each VR headset to capture the finger motion (Figure 3.5).

Figure 3.5 VR interface and simulated robot hand.

Test results In our tests, the goal of robot in-hand manipulation is to arbitrarily change the state of an object that moves on a plane as desired. Two position coordinates and one orientation coordinate of the object are used as the state variables. A Sobol sequence of 10^4 points is used to discretize the state space. 50 volunteers are invited to provide demonstrations using the VR interface. The human mentors are encouraged to control the robot hand to manipulate the object in as many ways as possible. Each mentor provides multiple demonstrations of different lengths, covering various areas and transitions in the state space. The demonstrations are processed and archived in the knowledge library as described earlier. Based on the crowdsourced knowledge library, new skills are generated to manipulate the object from various initial states to final states.

In order to examine the power of crowdsourcing, the synthesized skills are studied in two ways. First, the success rate of synthesizing new skills is studied with respect to the amount of information archived to the knowledge library. The robot hand is requested to synthesize a certain amount of random new skills when the archived demonstrations reach different percentages of coverage of the state space. The initial state and final state of new skills are randomly picked over the state space. Without

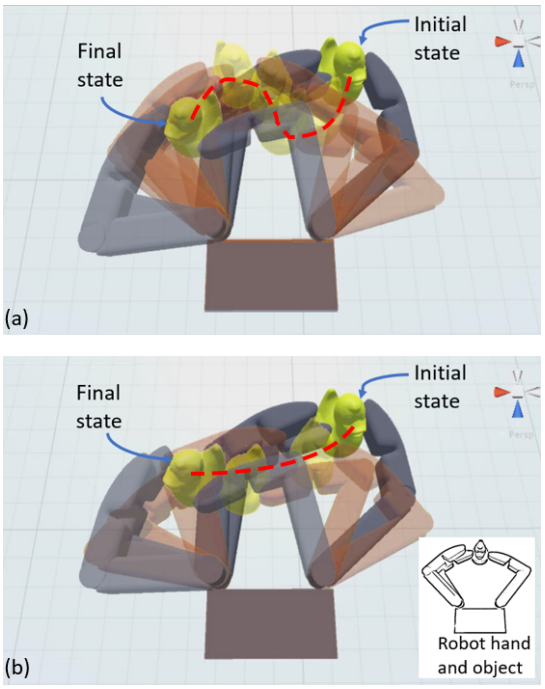

Figure 3.6 A skill synthesized using a knowledge library that covers (a) 21% of the state space, (b) 67% of the state space.

much surprise, we have observed that as more demonstrations are archived and their coverage of the state space increases, the success rate of skill synthesis increases about proportionally (Figure 3.7). This is mainly because there is little chance that a new skill can be synthesized with an initial state and/or final state that have never been covered by any demonstration.

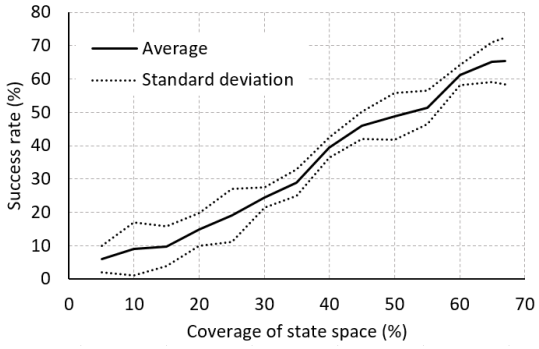

Figure 3.7 Success rate of new skill synthesis.

In addition, the performance of a successfully synthesized new skill is studied with respect to the total amount of demonstrations collected. Figure 3.6 shows the screenshots of the same skill (same initial state and same final state) synthesized when the amount of demonstrations collected reaches different levels. The first result is synthesized when 100 demonstrations are collected, covering 21% of the (discretized) state space. The second result is synthesized when 500 demonstration trajectories are collected, covering 67% of the (discretized) state space. Despite both being successful in terms of manipulating the object from the given initial state to the desired final state, the manipulation synthesized using a knowledge library with more coverage of the state space gives a much cleaner process that has little unnecessary maneuver.

3.4 Data-oriented State Space Discretization

The state space discretization method based on low-discrepancy pesudo-random sequences (Subsection 3.2.2) and the fidgeting test (Section 3.3) are motivated by the assumption that the robot system needs to visit all kinds of states over an "open" working region in the state space. Such an assumption is proper for some skills such as in-hand manipulation (e.g., fidgeting). Meanwhile, for many other skills, especially those involving the manipulation of multiple bodies, the working region of the robot in the state space usually spreads through highly irregular areas instead of a well-conditioned convex region (Figure 3.8). This makes a large portion of cells being practically empty when a uniform discretization is used. The issue is especially problematic for systems of high-dimensional state spaces. Our latest work develops a data-oriented discretization strategy that dynamically adjusts the cell allocation according to the incoming data so as to achieve efficient data representation [67].

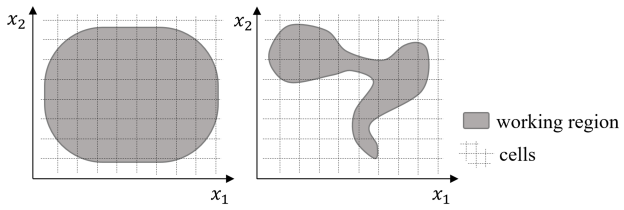

Figure 3.8 Open v.s. irregular working regions in the state space.

3.4.1 Dynamic cell allocation

Rather than determining the state space discretization in advance, the data-oriented method allocates cells dynamically according to the incoming data. First, an initial discretization is generated by allocating cells along the trajectory of one of the first demonstrations every certain data points. As new data come in, the same way is used to generate candidate new cells. Cell allocation is updated based on the comparison between the candidate new cells and the existing cells. A candidate new cell is adopted to the state space discretization if the difference is greater than a certain threshold. Otherwise, it is merged to existing cells that cover (mostly) the same area in the state space. Compared to a uniform discretization, the data-oriented strategy makes sure that the cells can represent the collected data efficiently. A (minor) disadvantage of the new strategy is the increasing total number of cells, which is fixed when using a pre-allocated discretization. Nevertheless, the issue is bearable - after a certain amount of data is collected, the chance of needing to add new cells becomes trivial. This is because there will have been enough cells to cover the system's working region in the state space.

3.4.2 Nearest neighbor search for data-oriented discretization

As explained in Subsection 3.2.2, most operations in the proposed learning scheme require either registering or locating a state at a cell in the discretized state space, which is in essence a nearest neighbor search problem. In addition to the non-aligned distribution, the data-oriented discretization strategy does not have the uniformity

property of the Sobol sequence. The exact solution of nearest neighbor search would require enumerating through the distances between the query point to all other points in the set. For large datasets obtained from crowdsourcing, such computation is not affordable. Search methods that give approximating solutions generally sit in four types which use search trees (e.g., KD-tree [68]), hashing (e.g., Locality Sensitive Hashing [69]), graphs (e.g., HNSW [70]), and quantization (e.g., Product Quantization [71]) respectively. According to experimental reports [72], graph-based methods tend to perform better than others.

Graph-based nearest neighbor search indexes the datasets to graphs. In our application, the update efficiency of the search graphs is of particular importance since the state space discretization can change frequently which causes new nodes being added to the search graph constantly. A high recall (i.e., precision), on the other hand is needed only when a candidate new cell is close to the existing cells. For ones that are far away from the existing cells, it is not really meaningful to find the exact nearest neighbor. This is because even the exact nearest neighbor is located, no effective control signal can be generated (via statistical inference) since the query point is too different from any data.

Based on the above considerations, a dual-graph search method is developed to provide nearest neighbor search for the data-oriented state space discretization method. Two directed graphs are used by the proposed search method. The nodes of both graphs correspond to the cells of the state space discretization (indexed by cell ID's) and the edges represent the neighboring relationships of the cells. Note that these graphs are constructed for the nearest neighbor search needed by the data-oriented state space discretization, and are different from the routing graph discussed in Subsection 3.2.3 for skill synthesis.

The first graph is called an In-Dataset Graph (IDG). This graph utilizes a particular search-and-build strategy (explained later) that guarantees to return the

query candidate cell itself if a same cell is already indexed in the graph (i.e., in-dataset query). In case that no existing cell overlaps with the candidate new cell, the IDG returns a relatively nearby existing cell. The result will then be used by the second graph, called a Diversified Graph (DG) as an initial point to search for a potentially closer neighbor. If the nearest neighbor found in the DG is too close (under a certain threshold) to the candidate cell, the candidate will be merged to it. Otherwise, the candidate new cell will be adopted to the state space discretization as well as indexed to the two search graphs. Note that the nodes in the IDG and DG are the same (both representing all the existing cells), whereas the edges of them are different. Both graphs are structured using "neighbor lists". In terms of topology, including a node in the neighbor list of another means the former is connected to the latter with an edge directing to it.

The two graphs are built and updated as the state space discretization changes when new data come in and candidate new cells are generated. Algorithm 2 explaines the search strategy for an IDG. The first node in an IDG is used as the initial base node. The distance between a query point and a base node is calculated as a baseline. Then, the nodes in the base node's neighbor list are checked in the order of when they were added to the graph. If a node in the list is found closer to the query point, it becomes a new base node. The search continues until all nodes in a neighbor list are checked and the base node is still the nearest to the query point. The last base node will be the returned search result from the IDG. In case of a candidate cell eventually getting adopted, the update strategy for an IDG is simply registering it as the (new) last node in the last searched neighbor list. A node newly added to the IDG has an empty neighbor list. Such a search-and-build strategy ensures that the adoption of a new node does not alter the search paths of all previously adopted nodes - because the nodes are always checked following the same orders in the neighbor lists. The feature guarantees the detection of the situation when a candidate new cell overlaps

with an existing cell, which leads to a merge. In other cases, if a query point is close to an existing node in an IDG, the search also has a great chance to locate that node.

Algorithm 2: Search using an IDG

```
1  IDG_SEARCH(IDG, q, k);
2  Parameters: the latest IDG (nodes {p_1, p_2, ...} and their neighbor lists),
     candidate new cell q, initial search node p_k;
3  Result: ID of the nearest node in IDG;
4  while i ≤ number_of_neighbors(p_k) do
5      n = ID of p_k's i'th neighbor;
6      if distance(q, p_n) < distance(q, p_k) then
7          k = n, i = 1;
8      else
9          i = i + 1;
10 Return k.
```

The result from searching over the IDG is passed to the DG as an initial node to search for a potentially closer neighbor of the candidate new cell. A greedy algorithm is used to search over a DG (Algorithm 3). Unlike searching over an IDG, all nodes in a neighbor list in a DG are checked, and the scale of neighbor lists has a major impact on the search efficiency. In order to limit the scale of the neighbor lists, the DG adopts a pruning strategy used in many graph-based search methods such as FANNG [73] and HNSW [70]. For a given node P_i and a distance metric d, a new node q can be adopted to P_i's neighbor list $L = \{L_1, L_2, ...\}$ if and only if $d(q, P_i) < min(d(q, L_1), ..., d(q, L_k))$, i.e., q is closer to P_i than it is to all nodes in the neighbor list of P_i. Refer to Figure 3.9 and consider a base node p whose neighbor list has two nodes a and b. A query point q is adopted to the neighbor list of p as a new node if and only if $d(q, p) < min(d(q, a), d(q, b))$. The other query point q' cannot

be adopted because $d(p, q') > d(a, q')$. Based on this rule, pruning is performed to truncate and bound the scale of the neighbor lists. Pruning has been known to promote the "diversification" of a graph [74], and gives the name of DG.

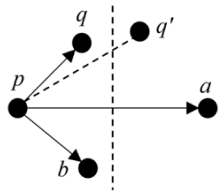

Figure 3.9 Construction of neighbor lists for DG.

Adding a new node to a complete DG requires enumeration - the new node should be first added to the DG with all other nodes in its neighbor list; then include the new node to all other nodes' neighbor lists; and finally perform pruning to all nodes' neighbor lists. The computational complexity for such an update procedure is O(nC), where n is the total number of nodes and C is the average out-degree (neighbor list size). For a large DG with a high update frequency required in our robot learning scheme, such a method is impractical. A reduced DG is constructed instead (Algorithm 4) to enable efficient DG updating. Although there is a sacrifice on the recall (precision), a strategy can be used to reduce the impact. Specifically, the search path of finding a candidate new cell's nearest neighbor using the DG is logged. When a new cell is added to the state space discretization and a corresponding node is added to the DG, its neighbor list will only include the nodes in the search path (instead of all other nodes). In the meantime, the new node will be adopted only to the neighbor lists of the nodes in the search path. If any neighbor list's size grows beyond the maximum allowed out-degree during the update, pruning will be performed using Algorithm 5.

In terms of computational efficiency, the monotonic search paths of an IDG give its search strategy a complexity of $O(log(n))$ [75], where n is the total number of nodes. For the search strategy of a DG, our simulation shows an empirical average

Algorithm 3: Search using a DG

```
DG_SEARCH(DG, p, q, k, m);
Parameters: the latest DG (nodes {p_1, p_2, ...} and their neighbor lists), a
    candidate new cell q, initial search node p_k, search memory m, candidate
    pool S = ∅, flag f = 0, search path log L = ∅;
Result: ID of the nearest neighbor, search path;
Add p_k to S, add p_k to L;
while f < m do
    f = the index of the first unchecked node in S;
    Mark S(f) as checked;
    for any neighbor p_n of p_k in DG do
        if p_n ∉ L then
            Add p_n to S;
            Add p_n to L;
    Sort the elements of S according to their distances to q in ascending
     order ;
    if size(S) > m then
        Truncate S to keep its first m elements;
Return j = ID of S(1), search path L.
```

Algorithm 4: Adopting a node to an IDG and a DG

```
REG_NEW_NODE(IDG, DG, q, C);
Parameters: the latest IDG and DG (nodes {p_1, p_2, ...} and their neighbor
    lists), candidate new cell q, initial search node p_k, search memory m,
    search path log L, threshold t, maximum out-degree R;
Result: updated IDG and DG;
k =IDG_SEARCH(IDG, q, k);
if distance(q, p_k) < t/2 then
    q is virtually the same as p_k. The candidate cell is not adopted;
else
    [j, L] = DG_SEARCH(DG, q, k, m);
    if distance(q, p_j) > t then
        Add q to the neighbor list of p_j in IDG;
        Add all nodes in L to the neighbor list of q in DG;
        Add q to the neighbor lists of all nodes in L in DG;
        for any node p_n ∈ L do
            if size(p_n's neighbor list) > R then
                DG_DIV(DG, p_n, R);
        if size(q's neighbor list) > R then
            DG_DIV(DG, q, R);
    else
        The candidate cell is not adopted.
```

Algorithm 5: Pruning for a DG

```
1  DG_DIV(DG, q, R);
2  Parameters: the latest DG (nodes {p_1, p_2, ...} and their neighbor lists),
     base node q, its neighbor list C before pruning, its neighbor list L = ∅ after
     pruning, maximum out-degree R;
3  Result: the selected neighbor list L;
4  Sort the neighbor list C in the ascending order of the distance to q;
5  for i = 1:size(C) do
6      for j = 1:size(L)+1 do
7          if j == size(L)+1 then
8              Add C(i) to L;
9          if distance(q, C(i)) > distance(q, L(j)) then
10             break.
11         if size(L)>R then
12             break.
```

complexity of O($Klog(n)$), where K is related to the search memory and the maximum out-degree. Such a complexity indicates that the reduced DG preserves the monotonic search character of a complete DG to some extent. The update strategy for an IDG takes trivial computation. The update strategy for a DG involves pruning, which is performed when the size of a neighbor list is greater than the maximum out-degree R. The total computational complexity for updating a DG is then bounded by $O(R^2log(R)Klog(n))$. Overall, the proposed dual-graph strategy gives a satisfactory balance between the recall and search/update efficiency.

3.5 Validation II - The Bottle Puzzle

3.5.1 Simulation

The data-oriented state space discretization is motivated to handle systems of irregular working regions in high-dimensional state spaces. A "bottle puzzle" that has a 12-dimensional state space is conceived to challenge the method. Figure 3.10 shows the setup of the test. The goal of the bottle puzzle is to use an T-shaped tool to retrieve a ball from inside a bottle. The bottle is fixed and has a tight opening, which makes the only possible way to retrieve the ball being a special maneuver that involves first managing to place the ball on top of the tool by bumping it against a corner. The 12 state variables of the system include the position and orientation of the T-shaped tool and the position and velocity of the ball. The control variables are the 6-axis force/torque applied to the tool. Other than having a high-dimensional state space, the bottle puzzle bears both "contemporary" and "sequential" characters. A skill is contemporary if no steps are involved and the control law is relatively time-invariant (e.g., balancing an inverted pendulum). Such skills often feature strong real-time dynamics. A skill is sequential if causal steps of specific sequences are needed whereas dynamical maneuvers are not necessarily involved (e.g., chess games). Skills of either of the two types have been well tackled separately by many learning methods. The

unique skill synthesis function of the proposed learning scheme is expected to allow robots to handle skills of both contemporary and sequential characters.

The test is first conducted using a simulator [67]. The physical dynamics is simulated using Unity Physics. Data management and robot learning are implemented using MATLAB. Similar to the simulation of the fidgeting test, the HTC Vive system is used to provide a virtual reality (VR) interface for human mentors to intuitively give demonstrations. The Unity Physics engine, MATLAB, and the VR interface are bridged using real-time UDP connections. Ten novice participants provided 50 demonstrations by controlling the (virtual) T-shaped tool using the HTC Vive hand-held controller. The control signals of the mentors and the response signals of the tool and the ball are collected as the demonstration data. Less than 8000 cells are created by the data-oriented discretization method. As a comparison, because of the high dimension of the state space, the method introduced in Subsection 3.2.2 using pseudo-random sequences can hardly provide satisfactory skill synthesis for this test even with more than 10^{20} cells.

Because of the challenging physical interactions involved, majority (43 out of 50) of the demonstrations are unsuccessful. Meanwhile, the few successful ones do not cover all possible situations. In particular, no successful demonstration has been collected with the ball initially under the tool, which is set to be the test case in the autonomous operation. Such a setting challenges the learning scheme on synthesizing new skills and reacting to unexpected situations that are never demonstrated or only partly demonstrated. Figure 3.13 shows a successful autonomous solving of the bottle puzzle. Due to the complicated physical contacts, the ball's response deviates from the expected trajectories easily if the control is applied in an open-loop manner (as opposed to real-time feedback control). Rather than blindly applying an entire synthesized control sequence from the beginning to the end, as the task goes on, it is necessary to conduct real-time skill re-synthesis to adjusts the control based on the

actual system response. Such a strategy provides closed-loop stability to some extent. Its successful implementation also approves the real-time computing efficiency of the proposed algorithms.

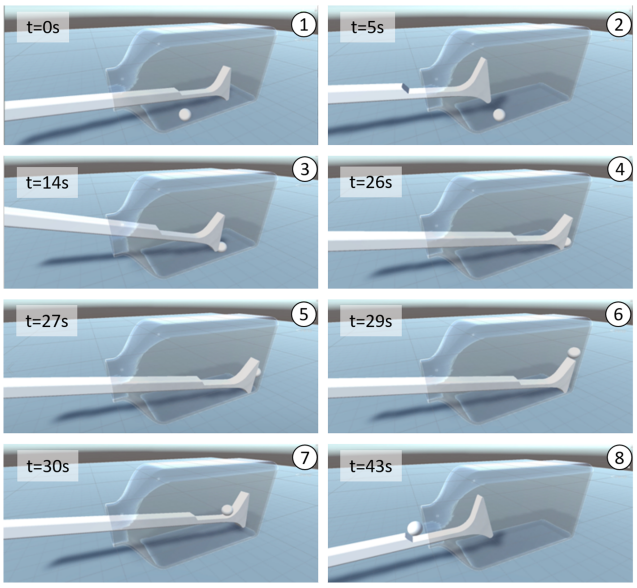

Figure 3.10 Successful autonomous solving of the bottle puzzle (simulated).

3.5.2 Physical tests

Our latest development includes a physical test of the bottle puzzle. In the simulated test, the force and torque applied on the tool are directly controlled. The physical test uses an AUBO i5 6-axis robot arm and a bionic robot hand [19] to manipulate the tool (Figure 3.11). Instead of rigidly mounting the tool to the robot, there is nontrivial looseness between the fingers of the robot hand and the tool. The slack grasping introduces additional degrees of freedom and causes significant passive movement of the tool with respect to the hand during the manipulation (Figure 3.11). This factor makes the task much more difficult than in the simulation, and is kept as a new challenge to test the potential of the proposed method.

In the physical test, the state of the system, including the motion variables of the T-shaped tool and the ball, is sensed by a vision system consisted of a high frame

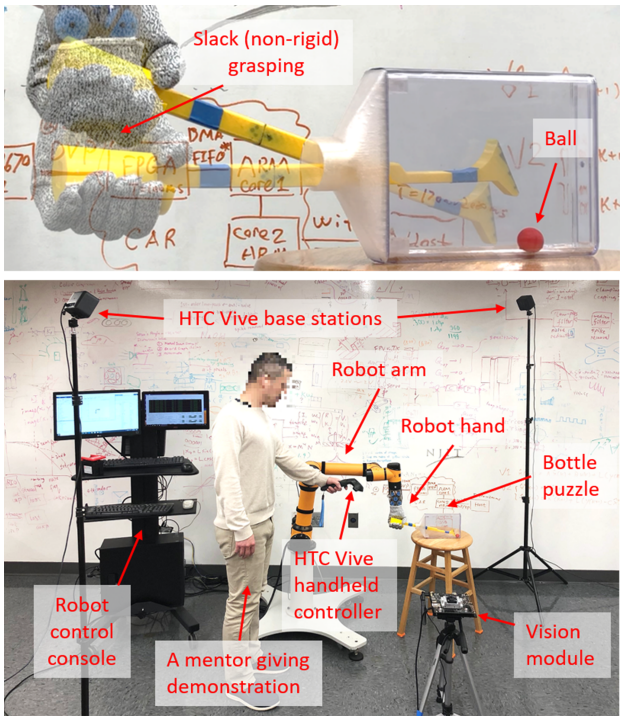

Figure 3.11 Physical setup of the bottle puzzle test.

rate camera as well as an embedded image acquisition and processing unit based on Nvidia's Jetson TX2 controller. Instead of shape recognition, color recognition is used to identify objects so as to shorten the sensing latency brought by image processing. In addition, instead of using simple differentiation to obtain velocity from position measurement, predictive learning [35] based on a dual-rate Kalman filter framework is used to enhance the accuracy of estimation and reduce negative effects brought by sensing latency. The vision acquisition and processing systems run on a standalone module which only sends the estimated state variables to the robot control console via UDP connections. The robot arm receives control signals via a Controller Area Network (CAN) bus. MATLAB is used to implement the learning scheme and generate control signals.

Unlike in the simulation, much more uncertain factors exist in the physical test, especially those caused by the slack grasping of the robot hand (which is kept as a major challenge to test the proposed method). The friction variation and other surface

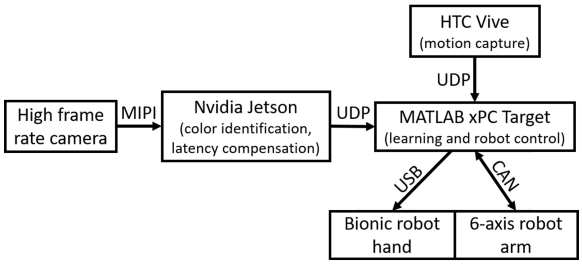

Figure 3.12 Sensing and control deployment of the physical test.

irregularities also contribute to the system uncertainty. These factors cause the system state to deviate from the expected trajectories and significantly lower the repeatability of actions. As a countermeasure, online skill re-synthesis is used to provide a form of state feedback control. In terms of implementation, the measures described earlier to shorten the sensing latency as well as the real-time capability of communication protocols all contribute to ensure the effectiveness of online skill re-synthesis.

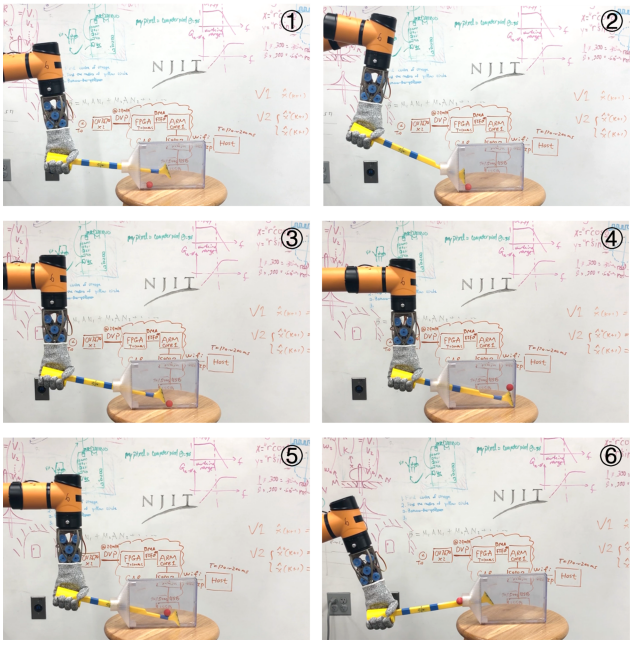

Figure 3.13 Successful autonomous solving of the bottle puzzle (physical test).

Demonstrations are provided by mentors through teleoperating the robot using the HTC Vive system. Again, a group of 10 novice human mentors provided 50 demonstrations. In order to make the operation as intuitive as possible, the motion

capture interface is configured to map the motion of the handheld controller directly to the robot hand using velocity tracking. Similar to the simulated test, in order to validate the synthesis of new skills, the initial conditions (i.e., the tool-ball relative positions) used in the autonomous tests are designed to be at those where no successful demonstrations have been provided. Figure 3.13 shows a synthesized successful autonomous execution of the task. The test results also show that the proposed learning scheme can handle strong uncertainties (e.g., the uncertain caused by the slack grasping).

3.6 Conclusions

We introduced a crowdsourced learning scheme for robots to acquire physical skills. The method allows robots to synthesize new physical skills using knowledge acquired from crowdsourced human mentors. It also provides a solution to sustainably manage a continuously growing massive knowledge library. Different methods are described for low dimension case and high dimension case. The proposed method formulates robot physical skills as transitions in the state space and the corresponding control actions. New skills are synthesized using a state space discretization technique and nonparametric statistical inference. The method has been validated using a simulated test of robot in-hand manipulation for the low dimension case. Virtual reality has been used to facilitate the validation with the participation of 50 human mentors. A study of the success rate of new skill synthesis and the performance of the synthesized skills has proved the effectiveness of crowdsourced learning. In order to handle systems with high-dimensional state spaces, a data-oriented state space discretization method is developed. The method dynamically allocates cells to discretize the state space based on the changing distribution of the collected data. A dual-graph nearest neighbor search algorithm is developed to realize computationally efficient dynamic

cell allocation. A simulated bottle puzzle experiment and physical experiment are discussed.

CHAPTER 4

CONCLUSIONS

In this dissertation, we mainly focus on the advanced robot physical skill learning and the ability of a robot to gain physical intelligence ubiquitously. For the first purpose, a composite robot learning scheme which learning from abstract definition and autonomous evaluation is proposed. Together, these complementary forms of robot learning allow humans to teach robots in an intuitive manner that is similar to teaching humans, where lecturing (of definitions), imitations (to demonstrations), and grading (of practicing trials) are integrated with school settings. The work aims for enabling robots to efficiently obtain advanced skills that have been difficult for them to learn so far, especially those requiring high dynamics, sophisticated contact control, accurate timing, and handling partly rigid partly soft gadgets. Adaptive approaches have been proposed to allow the robot to autonomously perceive human intentions in teaching through the demonstration data and evaluation. The "nunchaku flipping challenge" is introduced, which is an extreme test that puts hard requirements to all these three aspects.

For the second purpose, a unique learning scheme is developed to allow robots to learn from a group of mentors over long terms. State space discretization is used to sustainably manage constantly collected crowdsourced data and synthesize new skills. Two types of discretization methods are introduced. Initially, a discretization method based on pseudo random sequences is developed, featuring superior uniformity over the state space and a fixed upper bound of the archived data. Such a method is suitable for applications in which the robot system needs to visit all kinds of states over an open working region in the state space. For cases where the main working region of the robot in the state space spreads through highly irregular areas, a data-oriented state space discretization method is developed. The method aims

particularly at handling systems with high-dimensional state spaces. Simulation and physical tests of a fidgeting challenge and a bottle puzzle are conducted. The test results validate the proposed learning scheme's capability on synthesizing new skills, sustainable management of crowdsourced data, efficient handling of high-order systems, and tolerating strong system uncertainties.

REFERENCES

[1] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura, "The intelligent asimo: System overview and integration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, pp. 2478–2483, 2002.

[2] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter, "Bigdog, the rough-terrain quadruped robot," *International Federation of Automatic Control(IFAC) Proceedings Volumes*, vol. 41, no. 2, pp. 10822–10825, 2008.

[3] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson, "Optimization based full body control for the atlas robot," in *14th IEEE/RAS International Conference on Humanoid Robots*, pp. 120–127, 2014.

[4] Boston Dynamics: https://www.bostondynamics.com/handle, retrieved on February, 2017.

[5] S. Seok, A. Wang, M. Y. M. Chuah, D. J. Hyun, J. Lee, D. M. Otten, J. H. Lang, and S. Kim, "Design principles for energy-efficient legged locomotion and implementation on the mit cheetah robot," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 3, pp. 1117–1129, 2015.

[6] N. Furukawa, A. Namiki, S. Taku, and M. Ishikawa, "Dynamic regrasping using a high-speed multifingered hand and a high-speed vision system," in *IEEE International Conference on Robotics and Automation*, pp. 181–187, 2006.

[7] Y. Yamakawa, A. Namiki, M. Ishikawa, and M. Shimojo, "One-handed knotting of a flexible rope with a high-speed multifingered hand having tactile sensors," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 703–708, 2007.

[8] T. Ishihara, A. Namiki, M. Ishikawa, and M. Shimojo, "Dynamic pen spinning using a high-speed multifingered hand with high-speed tactile sensor," in *6th IEEE/RAS International Conference on Humanoid Robots*, pp. 258–263, 2006.

[9] A. G. Billard, S. Calinon, and R. Dillmann, "Learning from Humans," in *Springer Handbook of Robotics*, pp. 1995–2014, New York, NY: Springer, 2016.

[10] S. H. Huang, J. Pan, G. Mulcaire, and P. Abbeel, "Leveraging appearance priors in non-rigid registration, with application to manipulation of deformable objects," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 878–885, 2015.

[11] S. Calinon, P. Kormushev, and D. G. Caldwell, "Compliant skills acquisition and multi-optima policy search with em-based reinforcement learning," *Robotics and Autonomous Systems*, vol. 61, no. 4, pp. 369–379, 2013.

[12] J. Kober and J. Peters, "Imitation and reinforcement learning," *IEEE Robotics and Automation Magazine*, vol. 17, no. 2, pp. 55–62, 2010.

[13] S. Levine, N. Wagener, and P. Abbeel, "Learning contact-rich manipulation skills with guided policy search," in *IEEE International Conference on Robotics and Automation*, pp. 156–163, 2015.

[14] J. Fu, S. Levine, and P. Abbeel, "One-shot learning of manipulation skills with online dynamics adaptation and neural network priors," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4019–4026, 2016.

[15] M. P. Deisenroth, G. Neumann, J. Peters, *et al.*, "A survey on policy search for robotics," *Foundations and Trends in Robotics*, vol. 2, no. 1–2, pp. 1–142, 2013.

[16] Y. Huang, D. Büchler, O. Koç, B. Schölkopf, and J. Peters, "Jointly learning trajectory generation and hitting point prediction in robot table tennis," in *16th IEEE/RAS International Conference on Humanoid Robots*, pp. 650–655, 2016.

[17] J. Kober, A. Wilhelm, E. Oztop, and J. Peters, "Reinforcement learning to adjust parametrized motor primitives to new situations," *Autonomous Robots*, vol. 33, no. 4, pp. 361–379, 2012.

[18] L. Zhao, R. Lawhorn, S. Patil, S. Susanibar, B. Ouyang, L. Lu, and C. Wang, "Multiform adaptive robot skill learning from humans," in *American Society of Mechanical Engineers (ASME) Dynamic Systems and Control Conference*, October 2017.

[19] L. Zhao, Y. Zhao, S. Patil, D. Davies, C. Wang, L. Lu, and B. Ouyang, "Robot composite learning and the nunchaku flipping challenge," in *2018 IEEE International Conference on Robotics and Automation*, pp. 3160–3165, 2018.

[20] M. Cakmak and A. L. Thomaz, "Designing robot learners that ask good questions," in *the 7th Annual ACM/IEEE International Conference on Human-Robot Interaction*, pp. 17–24, 2012.

[21] B. Argall, B. Browning, and M. Veloso, "Learning by demonstration with critique from a human teacher," in *2nd ACM/IEEE International Conference on Human-Robot Interaction*, pp. 57–64, 2007.

[22] M. K. Johnson and L. Hasher, "Human learning and memory," *Annual Review of Psychology*, vol. 38, no. 1, pp. 631–668, 1987.

[23] C. L. Book, G. G. Duffy, L. R. Roehler, M. S. Meloth, and L. G. Vavrus, "A study of the relationship between teacher explanation and student metacognitive awareness during reading instruction," *Communication Education*, vol. 34, no. 1, pp. 29–36, 1985.

[24] T. J. Crooks, "The impact of classroom evaluation practices on students," *Review of Educational Research*, vol. 58, no. 4, pp. 438–481, 1988.

[25] S. Y. Chung and D. Y. Lee, "An augmented Petri net for modelling and control of assembly tasks with uncertainties," *International Journal of Computer Integrated Manufacturing*, vol. 18, no. 2-3, pp. 170–178, 2005.

[26] Q. Zhu, N. Wu, Y. Qiao, and M. Zhou, "Optimal scheduling of complex multi-cluster tools based on timed resource-oriented petri nets," *IEEE Access*, vol. 4, pp. 2096–2109, 2016.

[27] J. Mahler, M. Matl, X. Liu, A. Li, D. Gealy, and K. Goldberg, "Dex-net 3.0: Computing robust robot suction grasp targets in point clouds using a new analytic model and deep learning," *arXiv preprint arXiv:1709.06670*, 2017.

[28] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[29] D. Nguyen-Tuong and J. Peters, "Online kernel-based learning for task-space tracking robot control," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 9, pp. 1417–1425, 2012.

[30] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*, vol. 1. Cambridge, MA: MIT press Cambridge, 2006.

[31] C. Wang, Y. Zhao, C.-Y. Lin, and M. Tomizuka, "Fast planning of well conditioned trajectories for model learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1460–1465, September 2014.

[32] M. Gu and S. C. Eisenstat, "Efficient algorithms for computing a strong rank-revealing qr factorization," *SIAM Journal on Scientific Computing*, vol. 17, no. 4, pp. 848–869, 1996.

[33] Y. Tenzer, L. P. Jentoft, and R. D. Howe, "The feel of mems barometers: Inexpensive and easily customized tactile array sensors," *IEEE Robotics Automation Magazine*, vol. 21, pp. 89–95, Sept 2014.

[34] Q. Wang, G. Kurillo, F. Ofli, and R. Bajcsy, "Evaluation of pose tracking accuracy in the first and second generations of microsoft kinect," in *IEEE International Conference on Healthcare Informatics*, pp. 380–389, 2015.

[35] C. Wang, C.-Y. Lin, and M. Tomizuka, "Statistical learning algorithms to compensate slow visual feedback for industrial robots," *Journal of Dynamic Systems, Measurement, and Control*, vol. 137, no. 3, p. 031011, 2015.

[36] X. R. Li and V. P. Jilkov, "Survey of maneuvering target tracking. part i. dynamic models," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1333–1364, 2003.

[37] K. Tanaka, J. Parker, G. Baradoy, D. Sheehan, J. R. Holash, and L. Katz, "A comparison of exergaming interfaces for use in rehabilitation programs and research," *Loading...*, vol. 6, no. 9, 2012.

[38] C. Wang, W. Chen, and M. Tomizuka, "Robot end-effector sensing with position sensitive detector and inertial sensors," in *IEEE International Conference on Robotics and Automation*, pp. 5252–5257, May 2012.

[39] T. Senoo, A. Namiki, and M. Ishikawa, "High-speed batting using a multi-jointed manipulator," in *IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1191–1196, 2004.

[40] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[41] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge, MA: MIT press, 2018.

[42] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[43] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*, pp. 1928–1937, 2016.

[44] Will Knight: https://www.technologyreview.com/2018/07/30/240365, retrieved on July, 2018.

[45] J. Howe, *Crowdsourcing: Why the power of the crowd is driving the future of business*. New York, NY: Three River press, 2008.

[46] L. von Ahn, *Human Computation*. Dissertation, Carnegie Mellon University, Pittsburgh, PA, December 2005.

[47] A. J. Quinn and B. B. Bederson, "Human computation: a survey and taxonomy of a growing field," in *The SIGCHI Conference on Human Factors in Computing Systems*, pp. 1403–1412, ACM, 2011.

[48] A. Doan, R. Ramakrishnan, and A. Y. Halevy, "Crowdsourcing systems on the world-wide web," *Communications of the ACM*, vol. 54, no. 4, pp. 86–96, 2011.

[49] D. Geiger, S. Seedorf, T. Schulze, R. C. Nickerson, and M. Schader, "Managing the crowd: Towards a taxonomy of crowdsourcing processes," in *Americas Conference on Information Systems*, 2011.

[50] G. Little, L. B. Chilton, M. Goldman, and R. C. Miller, "Turkit: human computation algorithms on mechanical turk," in *The 23rd Annual ACM Symposium on User Interface Software and Technology*, pp. 57–66, 2010.

[51] J. I. Lipton, A. J. Fay, and D. Rus, "Baxter's homunculus: Virtual reality spaces for teleoperation in manufacturing," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 179–186, 2018.

[52] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, *et al.*, "Roboturk: A crowdsourcing platform for robotic skill learning through imitation," in *Proceedings of The 2nd Conference on Robot Learning*, vol. 87, pp. 879–893, 2018.

[53] A. Sorokin, D. Berenson, S. S. Srinivasa, and M. Hebert, "People helping robots helping people: Crowdsourcing for grasping novel objects," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2117–2122, October 2010.

[54] M. Forbes, M. Chung, M. Cakmak, and R. Rao, "Robot programming by demonstration with crowdsourced action fixes," in *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, vol. 2, 2014.

[55] M. J.-Y. Chung, M. Forbes, M. Cakmak, and R. P. Rao, "Accelerating imitation learning through crowdsourcing," in *2014 IEEE International Conference on Robotics and Automation*, pp. 4777–4784, 2014.

[56] I. S. Lee and H. Y. Lau, "Adaptive state space partitioning for reinforcement learning," *Engineering Applications of Artificial Intelligence*, vol. 17, no. 6, pp. 577–588, 2004.

[57] W. T. Uther and M. M. Veloso, "Tree based discretization for continuous state space reinforcement learning," in *The Annual Conference on Innovative Applications of Artificial Intelligence (IAAI)*, pp. 769–774, 1998.

[58] C. Wang, Y. Zhao, C.-Y. Lin, and M. Tomizuka, "Fast planning of well conditioned trajectories for model learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1460–1465, 2014.

[59] H. Niederreiter, "Low-discrepancy and low-dispersion sequences," *Journal of Number Theory*, vol. 30, no. 1, pp. 51–70, 1988.

[60] S. Joe and F. Y. Kuo, "Remark on algorithm 659: Implementing sobol's quasirandom sequence generator," *ACM Transactions on Mathematical Software*, vol. 29, no. 1, pp. 49–57, 2003.

[61] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[62] C. Wang, Y. Zhao, Y. Chen, and M. Tomizuka, "Nonparametric statistical learning control of robot manipulators for trajectory or contour tracking," *Robotics and Computer-Integrated Manufacturing*, vol. 35, pp. 96–103, 2015.

[63] R. Humphry, K. Jewell, and R. C. Rosenberger, "Development of in-hand manipulation and relationship with activities," *American Journal of Occupational Therapy*, vol. 49, no. 8, pp. 763–771, 1995.

[64] Amazon: https://www.seattletimes.com/business/amazon-warehouse-jobs-push-workers-to-physical-limit, retrieved on January, 2017.

[65] H. Van Hoof, T. Hermans, G. Neumann, and J. Peters, "Learning robot in-hand manipulation with tactile features," in *IEEE/RAS 15th International Conference on Humanoid Robots*, pp. 121–127, 2015.

[66] 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," in *2018 IEEE International Conference on Robotics and Automation*, pp. 5628–5635, 2018.

[67] L. Zhao, L. Lu, and C. Wang, "Data-oriented state space discretization for crowdsourced robot learning of physical skills," *American Society of Mechanical Engineers (ASME) Letters in Dynamic Systems and Control*, vol. 1, no. 2, 2020.

[68] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[69] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proceedings of the 25th International Conference on Very Large Data Bases*, pp. 518–529, 1999.

[70] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 4, pp. 824–836, 2018.

[71] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 117–128, 2011.

[72] Benchmark of ANN: https://github.com/erikbern/ann-benchmarks, retrieved on August, 2019.

[73] B. Harwood and T. Drummond, "Fanng: Fast approximate nearest neighbour graphs," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5713–5722, June 2016.

[74] W. Li, Y. Zhang, Y. Sun, W. Wang, M. Li, W. Zhang, and X. Lin, "Approximate nearest neighbor search on high dimensional data—experiments, analyses, and improvement," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 8, pp. 1475–1488, 2019.

[75] C. Fu, C. Xiang, C. Wang, and D. Cai, "Fast approximate nearest neighbor search with the navigating spreading-out graph," *Proceedings of the VLDB Endowment*, vol. 12, no. 5, pp. 461–474, 2019.