Journal of Intelligent & Robotic Systems (2022) 106:1
https://doi.org/10.1007/s10846-022-01680-7

SHORT PAPER

Multi-Phase Multi-Objective Dexterous Manipulation with Adaptive Hierarchical Curriculum

Lingfeng Tao[1] · Jiucai Zhang[2] · Xiaoli Zhang[1]

Received: 18 August 2021 / Accepted: 20 June 2022

Abstract
Dexterous manipulation tasks usually have multiple objectives. The priorities of these objectives may vary at different phases of a manipulation task. Current methods do not consider the objective priority and its change during the task, making a robot have a hard time or even fail to learn a good policy. In this work, we develop a novel Adaptive Hierarchical Curriculum to guide the robot to learn manipulation tasks with multiple prioritized objectives. Our method determines the objective priorities during the learning process and updates the learning sequence of the objectives to adapt to the changing priorities at different phases. A smooth transition function is developed to mitigate the effects on the learning stability when updating the learning sequence. The proposed method is validated in a multi-objective manipulation task with a JACO robot arm in which the robot needs to manipulate a target with obstacles surrounded. The simulation and physical experiment results show that the proposed method outperforms the baseline methods with a 92.5% success rate in 40 tests and on average takes 36.4% less time to finish the task.

Keywords Multi-phase multi-objective manipulation · Adaptive curriculum · Objective priority · Robot learning

1 Introduction

Dexterous manipulation is essential to increase robots' usability in assembly, healthcare, education, and living assistance. These tasks typically need to be finished in multiple phases, and each phase has multiple objectives [9, 10]. Although all phases usually share the same set of objectives [25, 30], the priorities of objectives in each phase can vary, which are critical to achieving the manipulation tasks' efficiency and success rate. For example, an assembly task usually has two phases: (1) approaching, (2) installation. All phases share three objectives: (a) fast speed, (b) high precision, and (c) avoid the collision. In the first phase, the robot picks up the assembly part and move to the target position. The task objective with the top priority is to avoid touching other parts, then try to move faster to minimize the execution time, and the lowest priority is to move precisely. In the second phase, the robot reaches the target position and is ready for installation, and now the priority order changes to high precision to improve the installation quality, minimize the execution time, and avoid touching other parts.

Existing research in the traditional control theory mainly focuses on weighing multiple objectives to balance objectives with optimization methods [13], which is computationally inefficient. Although deep reinforcement learning (DRL) has been proven effective in enabling the robot to conduct autonomous manipulation tasks intelligently [19], the current reward formulation is usually a linear summation of the reward components of objectives, which is implicit and inefficient to learn the objective priorities, and causing poor learning performance (i.e., take a long time to learn or even fail to learn a correct policy). Furthermore, the current reward mechanism is usually fixed through all phases. This one-fix-all solution (i.e., using the same objective priority for all phases) cannot ensure each phase's local performance to be optimal. Such solutions may lead to sub-optimal performance as the reward is not customized for each phase of

✉ Xiaoli Zhang
xlzhang@mines.edu

Lingfeng Tao
tao@mines.edu

Jiucai Zhang
zhangjiucai@gmail.com

[1] Colorado School of Mines, Intelligent Robotics and Systems Lab, 1500 Illinois St, Golden, CO 80401, USA

[2] GAC R&D Center Silicon Valley, Sunnyvale, CA 94085, USA

Published online: 16 August 2022

the task. As a result, the learning performance and learning efficiency are usually limited in manipulation tasks where multiple objectives exist.

In authors' vision, the most appropriate approach to achieve dexterous manipulation is Curriculum Learning [14]. Curriculum learning (CL) is getting popular in the RL domain to accelerate the training process by exploring how objectives can be prioritized into a curriculum to mitigate the difficulty of learning a problem from scratch. CL fundamentally provide a hierarchical sequence of objective, which is usually ordered from easy to difficult. Existing CL methods focus on building generalizable, transferable curricula to reduce the training efforts [17] in tasks like maze exploration and chess. To the authors' best knowledge, the study of CL in dexterous manipulation has been rarely reported.

One potential problem when adopting CL in robotics manipulation tasks is that the priority of multiple objectives is shared across the whole process of the task once it is determined. For a multi-phase manipulation task, share the same curricula across phases cannot guarantee an optimal task performance due to the objective priority difference in each phase. How to use the CL methods in such multi-phase tasks is an open problem.

In this work, we develop the Adaptive Hierarchical Curriculum (AHC) by considering objective priorities and priority changes across phases. Our method's novelty is that the robot with AHC can utilize the experience when exploring the environment to determine the objective priorities and update the learning sequence of the objectives based on the hierarchical curricula. The benefits include better learning efficiency and higher task performance. In summary, the main contributions of this work are as follow:

1) Propose Adaptive Hierarchical Curriculum to solve the manipulation task in a DRL manner to enable the robot to efficiently learn multiple objectives with different priorities, which changes in different task phases.
2) Validate the AHC method and compare different objective priority determination methods (i.e., autonomously determined dynamic objective priorities, empirically defined dynamic objective priorities, fixed objective priorities, and linear summation reward) in a multi-objective manipulation task.

2 Related Work

2.1 Traditional Manipulation Methods

Traditional non-learning control methods have tried to solve the manipulation task when multiple objectives need to be done. With a stochastic optimization procedure [13], the priorities of each objective can be automatically determined as the task changes. The priorities are represented as parameterized weight functions to adjust the linear summation of the output (e.g., force or torque) [22]. These methods develop multiple controllers for multiple objective functions and adjust controller outputs' contributions to take objective priorities into account. In contrast, our method allows the robot to use a single policy to embed the knowledge of objective priorities into the control strategies enabled by the DRL, which reduces the robot's computational burden once the policy is learned.

2.2 Learning-Based Manipulation Methods

DRL has been proven promising to enable autonomous manipulation, but the learning efficiency has been a great challenge. In [20], a DRL method combined off-policy updates and parallel training to reduce the training time and enable a physical robot to complete a door opening task. Kalashnikov et al. [7] utilized the human demonstration to initialize the DRL policy to effectively reduce the training time to control the robot hand in a hammer usage task and an object relocation task. For a multi-objective manipulation task, Haarnoja et al. [4] studied the compositionality of soft Q-learning methods in a multi-policy and multi-Q-function setup, which provides a method to construct new policies composing learned skills to improve learning efficiency. Despite the progress of existing research, learning efficiency in multi-objective dexterous manipulation tasks is still an open problem due to the difficulty of providing sufficient information for the reward design.

2.3 Curriculum Learning for Improved Learning Efficiency

Recent research proposes the CL method to improve the learning efficiency of RL training. CL seeks to speed up training by first training on a series of easier tasks and transfer the policy to the target tasks with a selected sequence [14]. In [3], A reverse curriculum generation method was proposed to gradually learn to reach the goal from a set of start states increasingly far from the goal, which leads to efficient training on goal-oriented tasks. A graph-based curriculum representation was proposed in [26] to automatically decide the fixed learning sequence of the objectives within the threshold of time. An adaptive curriculum was implemented in [16] to update the reward function during the training continuously. However, the adaptation is based on the decision of domain experts. Approaches that can automatically adjust the task sequence are proposed in [6, 15, 23], which enable the robot to adapt to the task priority change during learning. These approaches inspired the authors to adopt the automatic priority determination strategy to robotic manipulation tasks.

2.4 Reward Shaping to Incorporate Domain Knowledge

A similar approach that changes the reward function is the reward shaping method, which provides the robot with an extra reward based on the domain knowledge during the learning process to redirect the training [18]. Earlier approaches like [2] consistently provide an additional reward to the robot based on the state's potential. Later approaches start to discuss adaptive reward functions during the learning process. In [27], verbal feedback is used as an additional reward and occasionally provided to the robot when adapting to new conditions. A dynamic reward function with adjustable parameters is proposed in [5] to adjust the reward function based on experience. However, current dynamic reward functions typically use a linear summed reward, which does not contain a hierarchical structure to guide the robot to determine objective priorities.

3 Methodology

3.1 Model Structure

We model the manipulation task as an RL problem that follows the MDP [1]. The MDP is defined as a tuple $\{S, A, R, \gamma\}$, where $s \in S$ is the state of the environment, $a \in A$ is the set of actions. $R\left(s'|s,a\right)$ is the reward function to give the reward after the transition from state s to state s' with action a . γ is a discount factor. A policy $\pi(s,\theta)$ specifies the action for state s. θ is the policy network parameters. To find θ, Proximal Policy Optimization (PPO) algorithm [24] is adopted. It is a model-free, online, on-policy reinforcement learning algorithm. The PPO algorithm is a type of stochastic policy gradient descent approach that alternates between sampling data through environment interaction and optimizing a clipping surrogate objective function. To estimate the policy and value function, a PPO agent maintains two deep neural networks (DNN): 1) an actor $\pi(s,\theta)$, with parameters θ, takes observation s and returns the action command, 2) a critic $V(s,\phi)$, with parameters ϕ, takes observation s and returns the corresponding expectation of the discounted long-term reward. When training is complete, the trained optimal policy is stored in actor $\pi^*(s,\theta)$. DNN helps the DRL agent generalize knowledge from the observation instead of storing and looking up every state in traditional RL, reducing computational cost in tasks with large state and action space like robot control tasks. Compared to other DRL methods, PPO balances ease of implementation, sample complexity, and ease of tuning.

The curricula (reward function) is the design target of AHC. The task objectives and the task phases can be heuristically defined by the developers based on empirical knowledge and human perception. However, it is challenging for human experts to define appropriate curricula for each phase due to individual experience and bias. Thus, the robot should determine the priorities of objectives and generate curricula based on its own experience. For each objective, a reward component f_i is defined, where $i = 1,2,...,N$. N is the total number of the task's objectives. Each objective can only have one priority level, denoted as j, where $j = 1,2,...,N$. The priority level follows a descending manner as j increases (i.e., the objective with level 1 has the highest priority). The phases of the task are denoted as $k = 1,2,...,K$. K is the number of phases. The hierarchical curricula $\mathbb{R}^k$ for phase k is constructed with all reward components, which is introduced in the following section.

3.2 Hierarchical Curricula for Each Phase

The learning sequence of the hierarchical curricula in a phase k follows the same descending order as the objective priorities, which means the objectives with higher priorities are first learned. We define the curricula $\mathbb{R}^k = \left[R_1^k, ...R_j^k, ...R_N^k\right]$, which is a piecewise reward function based on the hierarchy. Similar strategies of designing piecewise reward functions can be found in [11, 32]. R_j^k is the curriculum (reward function) for curriculum level j (the same with objective priorities), which is computed as:

$$R_j^k = \sum_{m=1}^{j} f_i|m \qquad j = 1, 2, ..., N \tag{1}$$

where $f_i|m$ is the reward component for objective i with priority m . When the robot finishes learning a curriculum level j with a reward R_j^k, it will learn the next curriculum level by moving to the next reward component R_{j+1}^k. Such a reward function updating mechanism can be considered a reward shaping approach [18, 31] based on the objective priorities. In this work, the criterion of finish learning the curriculum level j in phase k is to check if the variance of the cumulative reward $\sum R_j^k$ for curriculum level j in the past 10 training episodes has converged to a threshold, which is empirically defined based on the actual reward function.

In practice, the above learning sequence is not strong enough to avoid interference with the higher-level priorities while learning the lower-level priorities (e.g., alter the previously learned policy). Thus, the learning process is subject to a hard priority constraint to avoid this interference. The criterion is that the reward components of learned higher-level objectives become constraints for the following episodes to avoid interference between curriculum levels. When learning an objective with curriculum level j , we define the constraints as:

$$\sum R^k_{j-1} \geq \tau_{j-1} \tag{2}$$

where threshold τ_{j-1} is the average of the episode reward $\sum R^k_{j-1}$ in the last 10 episodes. When Eq. 2 is violated, the current episode is terminated, and the collected experience of the current episode will not be saved in the experience pool. To prevent the constraint from being violated with the same agent, the policy will be updated with randomly sampled trajectories from the old experience. Then the next episode will start.

3.3 Determination of Objective Priorities in Each Phase

The goal of this section is to find the optimal hierarchical curricula for each phase. That is, to determine the priority of each objective. During the training, the robot is expected to explore all the phases of a given task and find the priorities and changes across different phases. However, at the start of the training, a single episode may not complete all the phases. One episode may end in an intermediate phase of the task due to failure or mistake. The robot needs sufficient information to determine objective priorities for a specific phase. We defined that the objective priorities for a phase can be determined and updated only when that phase is explored for P times, and P is the empirically defined threshold. Specifically, we count the number of times e_k that the robot visits a phase k, if $e_k \geq P$, the robot will determine the objective priorities of phase k based on the collected information. Before the priority determination, the robot will use a linear summed curricula of all reward components to guide its exploration.

In this work, we define the objective with a higher absolute average reward should be learned first, which means the corresponding objective is easier than others to collect reward no matter it is positive or negative. This strategy is based on traditional curriculum learning, where the curriculum can benefit when the training is ordered easy to difficult. In this work, the difficulty of an objective is defined as how hard to collect rewards. There can be other strategies to determine the objective priorities, such as the objectives with the least collected rewards should be learned first (difficult to easy), or the objectives with the lowest absolute rewards should be learned first (encourage exploration). We will study the effects of different strategies in future work.

We calculate the average episode reward of each objective:

$$\tilde{r}^k_i = \frac{1}{P}\sum_{p=1}^{P} R^k_{i,p} \tag{3}$$

where $R^k_{i,p} = \sum f_i|k,p$ is the episode reward for objective i in phase k at episode p . The priorities for the objectives are updated by sorting the average cumulative rewards. To ensure the sorting operation's veracity, the magnitude of the reward components should be consistent along the time horizon, which means that the reward components need to be normalized or be defined with the same principles. The details of the defined reward components in this work are shown in the next section. Then, we extract the index of the sorted absolute average episode reward for each objective:

$$\begin{aligned}&\left[I^k_1, ...I^k_j, ...I^k_N\right] = Index\left[sort_{high\rightarrow low}|\tilde{r}^k_i|\right]\\ &j = 1, 2, ..., N\end{aligned} \tag{4}$$

where I^k_j is the index of the $\tilde{r}^k_i$. Finally, the reward component of the priority j is assigned as f_{I_j}. That is:

$$f_i|j = f_{I_j}|j \tag{5}$$

We assume that the hierarchy will converge to a single optimal hierarchy for each phase with the defined strategy as the training goes on.

3.4 Smooth Transition of Reward Hierarchy across Phases

When the robot needs to adapt to the dynamic objective priorities in different phases, the current phase's curricula transits to a different one in the next phase. A smooth transition function is designed by continuously updating the curricula from one to another. It can mitigate the negative impact on the policy performance when curricula changes. For easy modification and symmetrical transition, monotonically bounded functions are better choices. In this work, we choose a *tanh* function as the smooth transition function:

$$tanh(x) = \frac{e^{2x}-1}{e^{2x}+1} \tag{6}$$

where x is the identification state input such as position or time. Then the curriculum becomes:

$$R^k_j = \begin{cases} \sum^j_{m=1} f_i|m\left[\frac{1}{2}tanh(\alpha x)\right] + \sum^j_{m=1} f_{i'}|m\left[\frac{1}{2}tanh(\alpha x)\right], & \text{if } x \leq \delta \\ \sum^j_{m=1} f_i|m, & \text{if } x > \delta \end{cases} \tag{7}$$

where i' is the updated objective index with priority level j after the phase change, i is the previous objective index with priority j before the phase change. α is the adjustable parameter to control the slope and range of the smooth function. In this work, α is 1. δ is the threshold of the transition period between phases, which can be empirically defined. It should be noted that δ is small enough compared to the whole state space so that the smooth transition

has minimal influence on the constraint identification and policy updating. x is calculated as:

$$x = \sqrt{\sum \left[s_t - b(k, k+1)\right]^2} \tag{8}$$

x the Euclidean distance between the current state s_t and the predefined phase boundary $b(k,k+1)$ between phase k and phase $k+1$.

4 Experiments

The experiment was first implemented in a simulated environment using V-REP [21] and a physical environment. Specifically, the task environment contains a JACO arm [12] with a cylindroid end effector attached to the end joint (Fig. 2, in the physical environment, the JACO arm just keeps its hand close for simplicity). The JACO arm system is composed of six inter-linked segments and five motors. It has six degrees of freedom (DOF), which refers to three DOF 6 movements in three-dimensional space (up, down, left, right, back, and forth) and three DOF, six movements of JACO's wrist (abduction, adduction, flexion, extension, pronation, and supination). The JACO arm has lightweight, durable materials, low power consumption, and a user-friendly interface, making it suitable for robotic research in the laboratory environment. In the experiment, the JACO arm is placed next to a table. The target is a cube (the length of the side is 0.2 m) on the top of the table and located at the center. The other objects on the table are obstacles. The robot's task is to use the end effector to push the target off the table without touching the obstacles. Three objectives are defined to be satisfied by the robot for optimal performance. The corresponding reward components are constructed in a binary form to ensure the consistency of sorting operation in eq. 4. To account for the state-action transition of the RL setting, the JACO arm works step by step with a step size of 0.05s. In each step, the JACO arm executes an action, moves to the next state, and receives a reward based on the reward function.

(1) Avoid obstacles:

$$f_1 = \begin{cases} -1, & \text{if } \sum V_{obstacle} \geq 0 \\ 1, & \text{if } \sum V_{obstacle} = 0 \end{cases} \tag{9}$$

where $V_{obstacle}$ is the absolute velocity of obstacles. The function returns a penalty -1 if the robot or target touches any of the obstacles. Otherwise, it returns 1.

(2) Manipulation:

$$f_2 = \begin{cases} -1, & \text{if } (d-l)_t \leq (d-l)_{t-1} \\ 1, & \text{if } (d-l)_t > (d-l)_{t-1} \end{cases} \tag{10}$$

where d is the distance of the target from the starting point, l is the distance between the end effector and the target center. The *if* statement compares d and l in current time step t and last time step $t-1$. Overall, f_2 encourages the robot to approach and push the target away from the initial position until it falls.

(3) Minimize the execution time:

$$f_3 = \begin{cases} -1, & \text{if } v_t < v_{t-1} \\ 1, & \text{if } v_t \geq v_{t-1} \end{cases} \tag{11}$$

where v is the absolute velocity of the end effector. The *if* statement compares v in current time step t and last time step $t-1$. This component encourages the robot to act quickly towards the target.

The constructed reward functions evaluate the task performance according to the robot's interaction with the environment, including the JACO arm and the end effector. The movement of the JACO arm and the end effector is a cumulative result of all joints. Thus, the PPO algorithm will learn a policy that distributively controls the JACO's joints to complete the desired movement which satisfies the task objectives by maximizing the total reward. In this work, the action space includes the first five joints of the JACO arm. Each joint works in position control mode (rotation angle) and takes an incremental command from three possible actions [-0.01, 0, 0.01] rad. Such setup ensures the control of the JACO will not exceed the maximum power limit (10A) and the maximum linear arm speed (15cm/s). The state observation includes *the joint angle, target position, target velocity, obstacle position, obstacle velocity, and the action of the last time step*. The PPO policy observes the state in each time step and outputs the action in a vector containing the five joints' target positions. The JACO arm will then simultaneously actuate the motors to control the effector to reach the target positions. All five joints are constrained to: 1) avoid damaging the robot such as hitting the table, 2) accelerate training by limiting the work space above the table. Each episode was set to 60 seconds to offer the robot enough time to explore the environment and finish the task. The deep neural network of the PPO agent has three fully connected layers, and each layer has 100 neurons. The network has one additional fully connected layer than the original PPO agent, and each layer has a few more neurons to handle the manipulation task. The hyper-parameters of PPO algorithms and AHC are presented in Table 1. The clip factor improves training stability by limiting the size of the policy change at each step with a clipped surrogate objective function. When

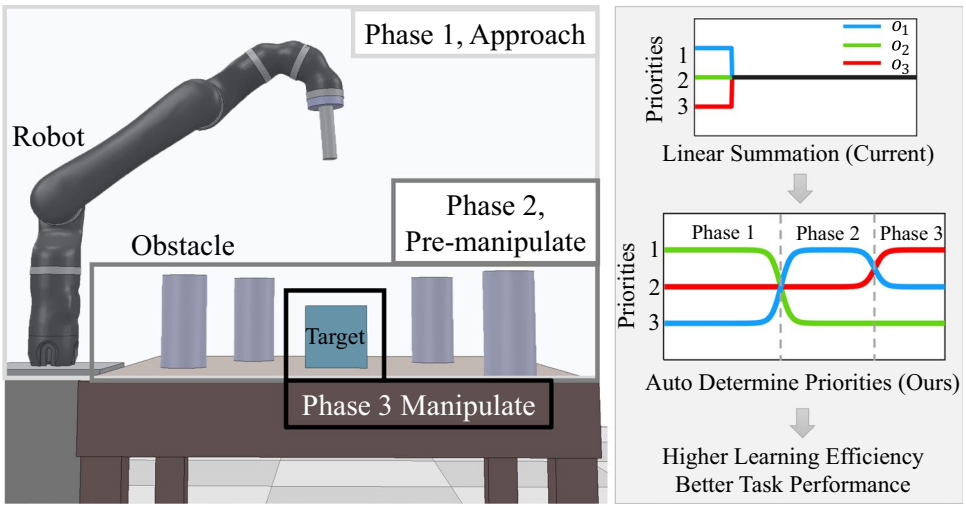

Fig. 1 A manipulation task with three phases: (1) approach; (2) pre-manipulate; (3) manipulate. Three task objectives are defined: avoid obstacles (o_1); minimize the execution time (o_2); manipulate (o_3). The objective priorities are expected to change in different phases. While a current reward mechanism usually uses linear summation, which is inefficient to learn and hard to find a good policy. In addition, the robot should determine the objective priorities during the learning process to improve its learning efficiency and policy performance

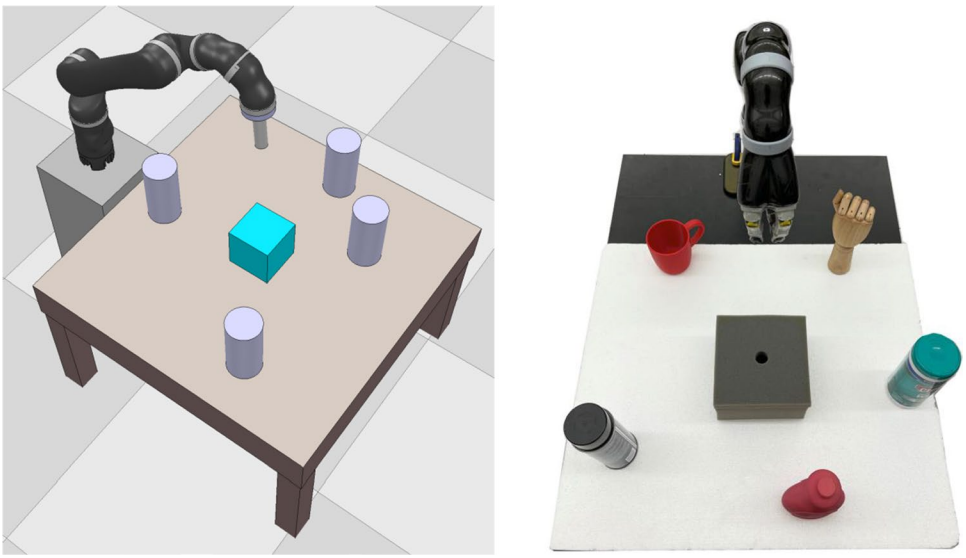

Fig. 2 Experiment setup. A JACO robot arm is attached with a cylindroid end effector next to a table. The target is a cube that is located at the center top of the table, the others as obstacles. The task is to push the target off the table and avoid touching the obstacles. The physical environment has a similar setup to mimic the simulation environment

Table 1 Hyper-Parameters for PPO Algorithm and AHC

Parameter	Value
Discount Factor (γ)	0.995
Experience Horizon	128
Entropy Loss Weight	0.02
Clip Factor	0.05
GAE Factor	0.95
Sample Time	0.05
Mini-Batch Size	64
Learning Rate	0.001
Number of Epoch	3
P	50
δ	0.05

computing the discounted sum of temporal difference errors, the Generalized Advantage Estimator (GAE) is a smoothing factor. The chosen of clip factor and GAE factor follows the default setup of the original PPO algorithm. The discount factor is set to 0.995 to constraint the cumulative reward to a finite horizon. The learning rate is set to 0.001, which must be considerably small to avoid large deviations during the parameter update process [28, 29]. The epoch number is set to 3 to utilize the latest experiences. Adam [8] optimizer was used for training the network. The experience horizon and mini-batch size is tuned based on the time steps of each episode. The sample time for the simulation is set to 0.05s because the simulated robot can execute the action

faster than the physical robot. After the training is finished in simulation, the trained policy was transferred to the physical environment for the test. The sample time was extended to 0.5s for the physical JACO arm to complete the action.

Based on the task environment, three phases are defined in spatial order (Fig. 1), and each phase covers a portion of the 3D task space. The phase identification is based on the position of the end effector. The first phase starts from the initial position and covers all the space outside of the obstacles' boundary. The second phase encompass the obstacles before touching the target cube. The top boundary of the second phase is 0.05m higher than the top surface of obstacles. The third phase is a cubic space around the target, whose side length is 0.25m.

Three other curricula are designed to evaluate the AHC method's performance. The first curricula are empirically defined (denoted as MHC), shown in Fig. 3b, with a different hierarchy in each phase. The second curricula are the same as the first one but without smooth transition function (denoted as MHC/NT), shown in Fig. 3c with different hierarchies in each phase. The third curricula use a fixed hierarchy across phases (denoted as FHC), shown in Fig. 3d. The baseline curricula use an equally weighted linear summation (denoted as LS), shown in Fig. 3e.

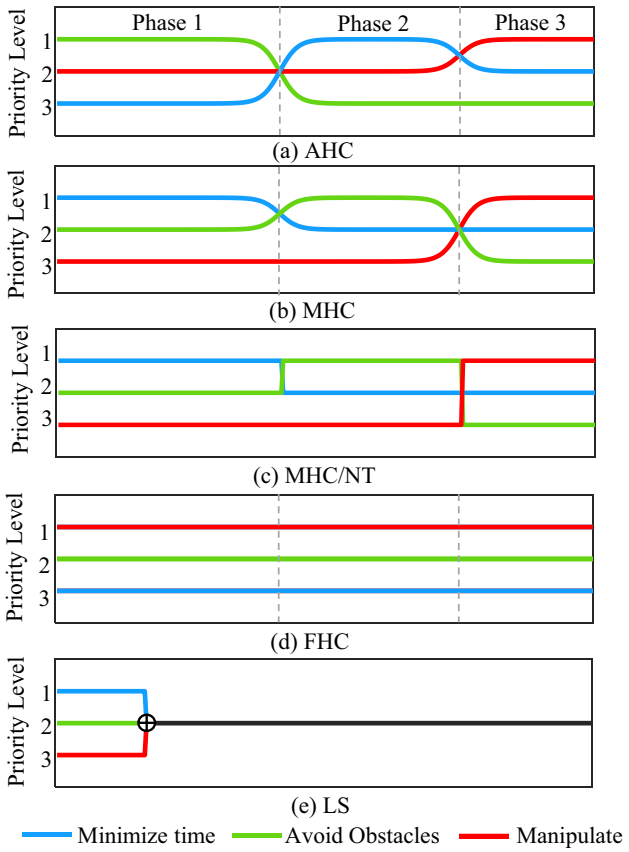

Fig. 3 **a** AHC: the objective priorities for each phase are determined by the robot. **b** MHC: the tasks priorities and their change for each phase are empirically defined. **c** MHC/NT: the tasks priorities are the same with MHC but without smooth transition. **d** FHC: the objective priorities are empirically defined and fixed. **e** LS: the reward is linearly summed

5 Results and Discussion

Figure 3a shows the learned objective priorities with AHC for each phase. All methods are trained 300 episodes 5 times. Figure 4 shows the normalized average reward and the standard deviation boundary during the training progress. It shows that the AHC method outperformed the other three methods. MHC converged faster than MHC/NT, which means the smooth transition function did contribute to the training. For AHC, the curricula for the three phases are determined after 50 episodes, 142 episodes, and 209 episodes. In phase 1, the convergence for the first 2 objectives occurred after 62 episodes, 113 episodes. In phase 2, the convergence for the first 2 objectives occurred after 161 episodes, 194 episodes. For phase 3, the convergence for the first 2 objectives occurred after 223 episodes, 261 episodes. The learning behavior shows that the objective with the top priority converges the fastest. The curricula for the next phase is usually determined after the second objective of the last phase converged.

The visualizations of the learned policies are shown in Figs. 5, 6 and 7. Each policy presents 6 screenshots show the critical moments in the task. Table 2 shows the performance statistics of the best performed trained policies. Comparing to MHC and MHC/NT, AHC determined different objective priorities in phase 1 and phase 2, shown in Fig. 3. In the experiment, the priority determination strategy is to learn the objective with a higher average reward. In phase 1, the robot can never touch

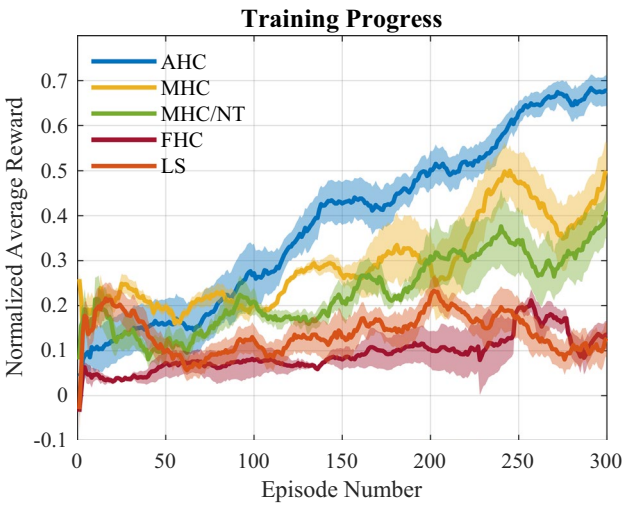

Fig. 4 Comparison of the learning progress for all the tested methods. All methods are trained 300 episodes 5 times

Table 2 Performance Statistics over 40 Evaluations

Method	S^*	S Rate	Touch Obstacles	Time (s)	Cube Travel Length (m)
AHC	37	92.5%	2	3.8 ± 0.3	0.5103 ± 0.0032
MHC	34	85%	5	4.4 ± 0.6	0.5148 ± 0.0024
MHC/NT	31	77.5%	9	4.7 ± 0.4	0.5125 ± 0.0029
FHC	30	75%	36	6.5 ± 1.3	0.5274 ± 0.0153
LS	4	10%	39	8.3 ± 1.1	0.2442 ± 0.0219

an obstacle, so the reward received for this objective is always the highest. Thus, avoid obstacles became the top priority. For phase 2, minimize time became the top priority. One possible explanation is that the robot spent more time adjusting its pose to prepare for manipulation, which brings a large penalty in execution time. Thus, the robot tried to transit to phase 3 at a faster speed to manipulate the target. The determined objective priorities are different from AHC and MHC (and MHC/NT). Our interpretation is that the priorities determined by AHC are optimal for learning the task, while the empirically defined objective priorities in MHC and MHC/NT match human developers' performance expectations, but it is sub-optimal for the robot to learn. FHC had difficulty maintaining the performance as the shared curricula cannot achieve local optimal for each phase. The robot with the LS method struggled to find the policy due to the implicit priority information. The results of AHC, MHC and MHC/NT in Table 2 prove that dynamically updating the objective priorities can effectively improve the learning efficiency and learning outcomes. Specifically, AHC achieved the best performance in objectives 1 and 3. For objective 2, the performances of AHC, MHC and MHC/NT are similar because they learned similar policies. The detailed performance statistics of the trained policies in different phases are shown in Table 3, which is included in the Appendix section.

The policy evaluation shows that different curricula can result in different robot behaviors, and such behaviors are consistent in multiple training, which means the learned behaviors are not dependent on the random exploration in early training. For the designed task, it is not easy to manipulate the target in any direction due to the end effector's cylindrical shape. Any inappropriate contact may move the target to an uncomfortable position, like rotate

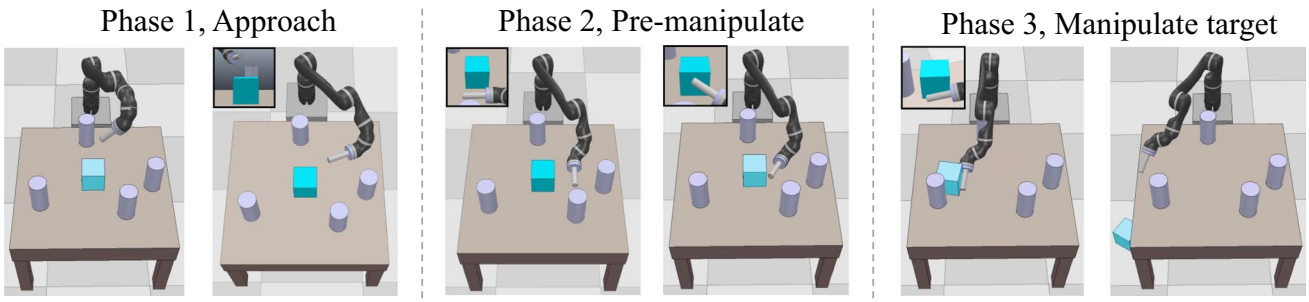

Fig. 5 Visualization of the AHC policy that has learned to push the target cube down the table and avoid touching the obstacles. Necessary magnified side views of the end-effector are included on the top-left corner of some keyframes to show the dexterity of the manipulation and how the robot achieve the objectives in each phases. The robot learned to approach the target from the right side, adjust the end-effector pose to prepare for manipulation, and then push the target down the table. The policies learned with the MHC method and MHC/NT method have similar behaviors but with a longer training time and worse task performance, as shown in Table I. For simplicity, the visualizations for MHC and MHCNT are not shown

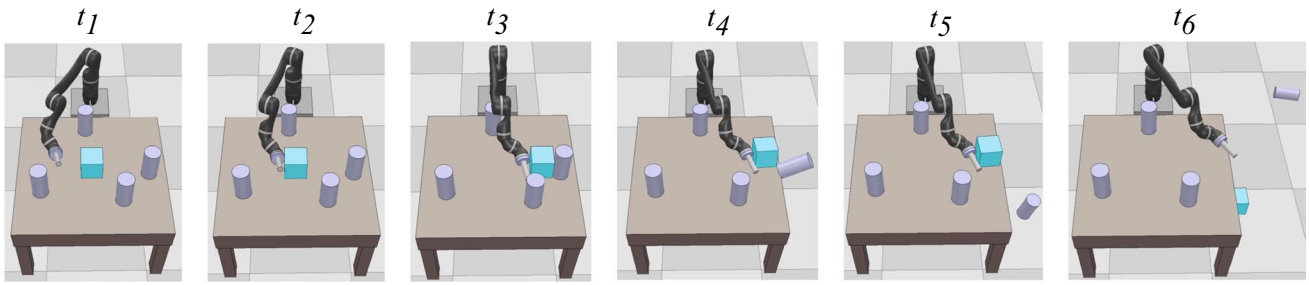

Fig. 6 Visualization of the FHC policy. The robot approached the target from the left side, then adjusted the pose and pushed down the target. Its behavior (i.e., approaching the target from the left side) is different from AHC and MHC, which has a higher chance to touch an obstacle and knocked it off

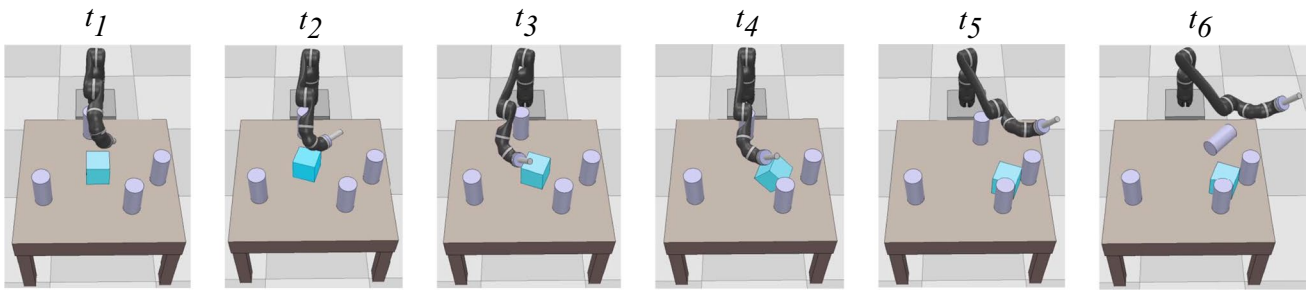

Fig. 7 Visualization of the LS policy. The robot approached the target from the top side. The end effector touched the target while it was adjusting pose, then caused the target to rotate before manipulating. As a result, the manipulation difficulty increased, and the robot failed the task

the target or push it away. The robot with the AHC learned that it should approach the target's side to align the end effector horizontally to maximize the contact surface and maintain stable contact points, as shown in Fig. 5. The robot also learned that approaching the right side of the target may achieve better performance even though the left side seems easier because there are fewer obstacles than the other side. Furthermore, the robot knows to slow down the action and make fine motion adjustments to contact the target. Finally, the robot manipulates the target faster as it already learned that push to the left has less chance to touch obstacles. The robot with the MHC and MHC/NT methods learned similar behaviors because the empirically defined objective priorities are near-optimal. The robot with FHC learned different behaviors to push the target to the right, as shown in Fig. 6. It can complete the task but has a higher chance to touch the obstacles. It is due to the fixed hierarchy cannot appropriately describe the objective priorities for all phases. As a result, the robot may only learn the top priority of the fixed hierarchy to push off the target but did not thoroughly learn the other two objectives. The LS method shown in Fig. 7 puts the robot's burden on finding the objective priorities during the learning process because the linear summed reward is abstract and implicit, which does not provide the information of objective priorities. As a result, the robot failed to find a stable policy in the learning process.

In this work, we focus on validating the proposed AHC approach in manipulation tasks involving multiple phases and objectives. Although the priority of objectives can be autonomously determined, to reduce implementation effort in the experiment, the number of the phases and phase transition time are empirically defined by domain experts. Thus, the designed phases and phase transition time may not be optimal and can be biased by individual knowledge. Our future work will study the autonomous phase recognition methods and task decomposition methods which can be integrated into AHC to increase the autonomy and intelligence of the robot in a manipulation task.

6 Conclusion

This work proposes an Adaptive Hierarchical Reward Mechanism method to help the robot determine and adapt to the objective priority changes while learning a multi-objective manipulation task. The experiment results show that different objective priorities can greatly affect the robot's performance. In a multi-objective manipulation task, it is essential to enable the robot to recognize the appropriate learning sequence of the multiple objectives and adapt to them for better task performance during the manipulation process. Our future work will focus on improving the autonomy and intelligence of the robot with automatic phase segmentation task decomposition.

Appendix:

Table 3 shows the performance statistics of the trained policies across the three phases over 40 evaluations, including the number of times that the robot touched obstacles, average time consumption in each phase, and the average normalized reward in each phase. LS is not included because it has no phases during the training.

Table 3 Performance Statistics across Phases over 40 Evaluations

Method	Phase	Touch Obstacles	Time (s)	Normalized Reward
AHC	*I*	0	0.9 ± 0.2	0.13 ± 0.08
	II	1	1.3 ± 0.3	0.29 ± 0.13
	III	1	1.4 ± 0.1	0.62 ± 0.21
MHC	*I*	0	1.2 ± 0.3	0.12 ± 0.04
	II	4	1.9 ± 0.3	0.25 ± 0.09
	III	1	1.5 ± 0.2	0.60 ± 0.16
MHC/NT	*I*	0	1.3 ± 0.3	0.15 ± 0.07
	II	7	2.1 ± 0.2	0.21 ± 0.11
	III	2	1.4 ± 0.4	0.52 ± 0.16
FHC	*I*	0	0.7 ± 0.2	0.19 ± 0.05
	II	5	1.7 ± 0.4	0.27 ± 0.08
	III	31	5.8 ± 1.3	0.37 ± 0.17

Author Contributions All authors contributed to the study conception and design. The first manuscript was written by Lingfeng Tao. Dr. Jiucai Zhang and Dr. Xiaoli Zhang provided comments and edits towards the creation of the final manuscript.

Funding This material is based on work supported by the US NSF under grant 1652454 and 2114464. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the National Science Foundation.

Data Availability Not applicable.

Code Availability Not applicable.

Declarations

Ethics Approval Ethical approval was waived by the local Ethics Committee of Colorado Schoolof Mines in view of the retrospective nature of the study and all the proceduresbeing performed were part of the routine care.

Consents to Participate Informed consent was obtained from all individual participants included in the study.

Consents for Publication The participants have consented to the submission of the case report to the journal.

Conflict of Interests Not applicable.

References

1. Bellman, R.: A markovian decision process. J Appl Math Mech. **6**(5), 679–684 (1957)
2. Devlin, S., Kudenko, D., Grześ, M.: An empirical study of potential-based reward shaping and advice in complex, multi-agent systems. Adv. Complex Syst. **14**(2), 251–278 (2011)
3. Florensa, C., Held, D., Wulfmeier, M., Zhang, M., Abbeel, P.: Reverse curriculum generation for reinforcement learning. In: Conference on robot learning, pp. 482–495. PMLR (2017)
4. Haarnoja, T., Pong, V., Zhou, A., Dalal, M., Abbeel, P., Levine, S.: Composable deep reinforcement learning for robotic manipulation. In: 2018 IEEE international conference on robotics and automation (ICRA), pp. 6244–6251. IEEE (2018)
5. Hu, Z., Kaifang, W., Gao, X., Zhai, Y.: A dynamic adjusting reward function method for deep reinforcement learning with adjustable parameters. Math Probl. Eng. 1–10 (2019). https://doi.org/10.1155/2019/7619483
6. Jain V, Tulabandhula T (2017) Faster Reinforcement learning using active simulators. CoRR. abs/1703.07853. http://arxiv.org/abs/1703.07853
7. Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., et al.: Scalable deep reinforcement learning for vision-based robotic manipulation. In: Conference on robot learning, pp. 651–673. PMLR (2018)
8. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) 2015 3rd International Conference on Learning Representations, (ICLR). Conference Track Proceedings, San Diego (2015). http://arxiv.org/abs/1412.6980
9. Kroemer, O., Daniel, C., Neumann, G., Van Hoof, H., Herke, G., Peters, J.: Towards learning hierarchical skills for multi-phase manipulation tasks. In: 2015 IEEE international conference on robotics and automation (ICRA), pp. 1503–1510. IEEE (2015)
10. Kroemer, O., Van Hoof, H., Neumann, G., Peters, J.: Learning to predict phases of manipulation tasks as hidden states. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 4009–4014. IEEE (2014)
11. Luo, Y., Dong, K., Zhao, L., Sun, Z., Zhou, C., Song, B.: Balance between efficient and effective learning: Dense2sparse reward shaping for robot manipulation with environment uncertainty. arXiv preprint arXiv:2003.02740. (2020)
12. Maheu, V., Archambault, P.S., Frappier, J., Routhier, F.: Evaluation of the JACO robotic arm: Clinico-economic study for powered wheelchair users with upper-extremity disabilities. In: 2011 IEEE international conference on rehabilitation robotics, pp. 1–5. IEEE (2011)
13. Modugno, V., Neumann, G., Rueckert, E., Oriolo, G., Peters, J., Ivaldi, S.: Learning soft task priorities for control of redundant robots. In: 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 21–226. IEEE (2016)
14. Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M.E., Stone, P.: Curriculum learning for reinforcement learning domains: a framework and survey. CoRR. abs/2003.04960 (2020) https://arxiv.org/abs/2003.04960
15. Narvekar, S., Sinapov, J., Stone, P.: Autonomous task sequencing for customized curriculum design in reinforcement learning, pp. 2536–2542 (2017) IJCAI
16. Narvekar, S., Stone, P.: Learning curriculum policies for reinforcement learning. CoRR. abs/1812.00285 (2018) http://arxiv.org/abs/1812.00285
17. Narvekar, S., Stone, P.: Generalizing curricula for reinforcement learning. In: 2020 4th lifelong machine learning workshop at ICML (2020) https://openreview.net/forum?id=7YCysi_070N
18. Ng, A.Y., Harada, D., Russell, S.J.: Policy invariance under reward transformations: Theory and application to reward shaping. Icml. **99**, 278–287 (1999)
19. Nguyen, H., La, H.: Review of deep reinforcement learning for robot manipulation. In: 2019 Third IEEE International Conference on Robotic Computing (IRC), pp. 590–595. IEEE (2019)
20. Popov, I., Heess, N., Lillicrap, T.P., Hafner, R., Barth-Maron, G., Vecerik, M., Lampe, T., Tassa, Y., Erez, T., Riedmiller, M.A.: Data-efficient deep reinforcement learning for dexterous manipulation. CoRR. abs/1704.03073 (2017) http://arxiv.org/abs/1704.03073
21. Rohmer, E., Singh, S.S.P., Freese, M.: V-REP: A versatile and scalable robot simulation framework. In: 2013 IEEE/RSJ international conference on intelligent robots and systems, pp. 1321–1326. IEEE (2013)
22. Salini, J., Padois, V., Bidaud, P.: Synthesis of complex humanoid whole-body behavior: A focus on sequencing and tasks transitions. In: 2011 IEEE international conference on robotics and automation, pp. 1283–1290. IEEE (2011)
23. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized experience replay. In: Bengio, Y., Lecun, Y. (eds.) 2016 4th International Conference on Learning Representations, (ICLR). Conference Track Proceedings, San Juan (2016) http://arxiv.org/abs/1511.05952
24. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. CoRR. abs/1707.06347 (2017) http://arxiv.org/abs/1707.06347
25. Sharma, M., Liang, J., Zhao, J., LaGrassa, A., Kroemer, O.: Learning to compose hierarchical object-centric controllers for robotic manipulation. CoRR. abs/2011.04627 (2020) https://arxiv.org/abs/2011.04627
26. Da Silva, F.L., Da Costa, L., Reali, A.H.: Object-oriented curriculum generation for reinforcement learning. In: Proceedings

of the 17th international conference on autonomous agents and multiagent systems, pp. 1026–1034 (2018)
27. Tenorio-Gonzalez, A.C., Morales, E.F., Villasenor-Pineda, L.: Dynamic reward shaping: training a robot by voice. In: Ibero-American conference on artificial intelligence, pp. 483–492. Springer (2010)
28. Tutsoy, O., Barkana, D.E., Tugal, H.: Design of a completely model free adaptive control in the presence of parametric, non-parametric uncertainties and random control signal delay. ISA Trans. **76**, 67–77 (2018)
29. Tutsoy, O., Erol Barkana, D., Sule, C.: Learning to balance an NAO robot using reinforcement learning with symbolic inverse kinematic. Trans. Inst. Meas. Control. **39**(11), 1735–1748 (2017)
30. Veiga, F., Akrour, R., Peters, J.: Hierarchical tactile-based control decomposition of dexterous in-hand manipulation tasks. Front Robot AI. **7**. https://www.frontiersin.org/article/10.3389/frobt.2020.521448, (2020). https://doi.org/10.3389/frobt.2020.521448
31. Zhang, D., Bailey, C.P.: Obstacle avoidance and navigation utilizing reinforcement learning with reward shaping. In: Pham, T., Solomon, L., Rainey, K. (eds.) Artificial intelligence and machine learning for multi-domain operations applications II, vol. 11413, pp. 500–506. International Society for Optics and Photonics (SPIE) (2020). https://doi.org/10.1117/12.2558212
32. Zhu, Y., Wang, Z., Merel, J., Rusu, A.A., Erez, T., Cabi, S., Tunyasuvunakool, S., Kramár, J., Hadsell, R., de Freitas, N., Heess, N.: Reinforcement and imitation learning for diverse visuomotor skills. CoRR. abs/1802.09564 (2018) http://arxiv.org/abs/1802.09564

Lingfeng Tao received B.S. degrees in both Mechanical Engineering and Aerospace Engineering from SUNY, University at Buffalo in 2017, an M.S. degree in Mechanical Engineering from University of Pittsburgh in 2018. He is currently a Ph.D. candidate in the Department of Mechanical Engineering at Colorado School of Mines in Golden, CO, USA. His research interests include human-robot interaction and cooperation, reinforcement learning, shared control, and telemanipulation.

Jiucai Zhang is a chief architect at GAC R&D Center in Silicon Valley. He got his Ph.D. in computer engineering from the University of Nebraska Lincoln in 2011. His research focuses on machine learning, autonomous vehicles, and smart mobility. Prior to joining GAC R&D Center in Silicon Valley, he designed and developed algorithms and software for smart mobility and connected autonomous electric vehicles at National Renewable Energy Laboratory, General Electric and A123 Systems, Inc.

Xiaoli Zhang received a B.S. degree in Mechanical and Automation Engineering and an M.S. degree in Mechatronics Engineering from Xi'an Jiaotong University in Xi'an, China, in 2003 and 2006, respectively, and a Ph.D. degree in Biomedical Engineering from the University of Nebraska-Lincoln in Lincoln, NE, USA, in 2009. She is currently an Associate Professor with the Department of Mechanical Engineering at Colorado School of Mines in Golden, CO, USA. Her research interests include intelligent human–robot interaction and cooperation, human intention awareness, data-driven modeling, prediction, and control.