Policy Blending and Recombination for Multimodal Contact-Rich Tasks

Tetsuya Narita[1] and Oliver Kroemer[2]

***Abstract*— Multimodal information such as tactile, proximity and force sensing is essential for performing stable contact-rich manipulations. However, coupling multimodal information and motion control still remains a challenging topic. Rather than learning a monolithic skill policy that takes in all feedback signals at all times, skills should be divided into phases and learn to only use the sensor signals applicable to that phase. This makes learning the primitive policies for each phase easier, and allows the primitive policies to be more easily reused among different skills. However, stopping and abruptly switching between each primitive policy results in longer execution times and less robust behaviours. We therefore propose a blending approach to seamlessly combining the primitive policies into a reliable combined control policy. We evaluate both time-based and state-based blending approaches. The resulting approach was successfully evaluated in simulation and on a real robot, with an augmented finger vision sensor, on: opening a cap, turning a dial and flipping a breaker tasks. The evaluations show that the blended policies with multimodal feedback can be easily learned and reliably executed.**

I. INTRODUCTION

Manipulation skills that enable robots to handle varied objects and perform a wide range of tasks are in high demand. This is especially true for contact-rich tasks that require precise motions and gentle contacts for stable interactions with the environment. Such manipulation skills can often be divided into multiple phases [1]. For example, a grasping skill can be decomposed into phases corresponding to reaching towards an object, closing the gripper, and lifting up the object. Sensor feedback provides crucial information for monitoring the task execution and achieving accurate control in each manipulation phase. Since the purpose of each manipulation phase is different, the required sensor feedback also varies across phases. In particular, vision, proximity, haptic, and tactile information are all useful for contact-rich tasks, with the former being more useful for establishing contacts and the latter for exploiting contacts.

A key challenge to realizing multimodal contact-rich tasks is coupling the sensor modalities with the controller. Conventional rule-based approaches often require manually designed control algorithms for combining sensor modalities [10], [18]. Such approaches require fine-tuning of control parameters with consideration of the influence from other modalities and suffer from environment changes. Recent works have developed data driven approaches [19], [32]. These approaches have mainly focused on representing control policies with

[1]Tetsuya Narita was with the Carnegie Mellon University. He is now with Sony Corporation, Tokyo 141-8610, Japan (e-mail: Tetsuya.A.Narita@sony.com).

[2]Oliver Kroemer is with Robotics Institute, Carnegie Mellon University, Pittsburgh, PA USA (e-mail: okroemer@andrew.cmu.edu).

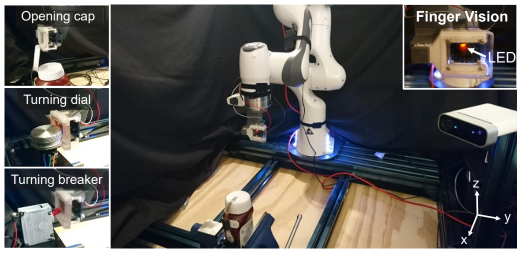

Fig. 1. Overview of the real environment. Finger vision tactile sensor is attached at the end-effector. Red LED is used to detect only near object surface. Three different tasks `opening cap`, `turning dial`, and `turning breaker` is setup for evaluation.

neural networks or graph structures which are trained from large amounts of data. Reinforcement Learning (RL) is an effective approach to training such policies from experience.

However, these data-driven approaches often represent skills in a monolithic manner, e.g. as one large neural network, rather than adopting a phase-like structure. As a result, the skills need to learn to utilize the full set of sensor signals at each time, including when outside of the sensor's operating range. Decomposing a skill into a sequence of phases allows the robot to learn separate feedback controllers for each phase and it provides modularity for reusing phases between skills. The disadvantage of this modular phase-based approach is that naive sequential executions can often result in worse performance. Stopping between each phase and abruptly switching between sensor modalities can lead to longer execution times and less robust behaviours. Therefore, each phase's primitive policy should be blended into the next one to create a seamless control policy.

This paper proposes a policy blending approach to address the above problems. We first train the robot to learn a set of primitive policies coupled with specific sensor modalities. The robot subsequently learns a blending policy for overlapping and combining the primitive policies into a seamless control policy for performing the task. We evaluate both time-dependent and state-dependent blending policies. This policy blending approach has the following key benefits: 1) Each primitive policy is easy to train. 2) The blending strategy produces continuous trajectories between different policies. 3) The blending policy and initial primitives can be reused to create similar firm contacts for different manipulation tasks by replacing the final primitive.

The proposed approach was evaluated in both simulation and on a real robot (Fig.1). Three target tasks were evaluated: `opening cap`, `turning dial` and `turning breaker`. The experiments evaluated the effectiveness of the multimodal sensory feedback and the blending approach. The results shows that the target tasks could be performed faster and more reliably by using the blending approach.

II. RELATED WORK

A. *Rule-based multimodal manipulation*

Early rule-based approaches to handling multimodal information [10], [11] focused on combining vision and tactile sensory information for grasping tasks. [12] designs controllers for three modalities (position, vision, tactile), and combined them into an impedance force controller. [14] proposes a hierarchical structure for tactile and visual servo. [17] demonstrates exploratory behaviors of objects using tactile and vision sensor. Although these approaches are promising to perform particular tasks, significant effort for designing and tuning controllers for each modality is needed. Also, these approaches often need to be applied to diverse environments and objects, which makes designing a controller difficult. Multimodal information is also used for estimating the state of objects for manipulation. For example, tactile sensors are often used as compensation for missing vision information such as occlusion. Researchers have combined tactile and vision based approaches to estimate precise object models [15], [16], [18]. [13] estimates dynamic motion of rigid body when the robot's fingers touch an object.

B. *Data-driven multimodal manipulation*

Data driven approaches, such as supervised or reinforcement learning, have recently been proposed to handle various objects. Multimodal information is frequently used for object recognition or contact perception network. [26] uses force, thermal and motion data to train neural networks for rich contact perception. [25] and [30] combine RGB and depth information for better object recognition. However, coupling recognition and control still remains an open research area. End-to-end learning is one approach to coupling recognition of the environment and robot control. [23], [24] propose a multimodal network which maps from sensor values to robot actions. [31] uses feed-forward neural networks, and [32] uses self-supervise learning for grasping. Trained networks can also be used for anomaly detection for manipulation tasks [28]. Demonstration data is another approach to train multimodal networks. [27] uses sensor gloves to train grasp policies and [29] uses a game controller for demonstration. Although end-to-end approaches have confirmed the effectiveness of leveraging multimodal sensing for control, they usually lead to large state spaces, which makes it difficult to get convergence within practical time limits.

Latent representations is one possible solution to avoid the expansion of state space dimension. This approach is also effective to acquiring generalization between different object configurations, environments, and tasks. [21] proposes neural networks of multimodal representations to realize peg insertion that is transferable to different peg shapes. Latent representations can also handle verbal information. [19] maps natural language, point-cloud and trajectory information with deep multimodal embedding space. [20] uses representations of audio and video inputs for speech classification. [22] predicts temporal sequence of different manipulation tasks with auto-encoder network. However, generalizing representations

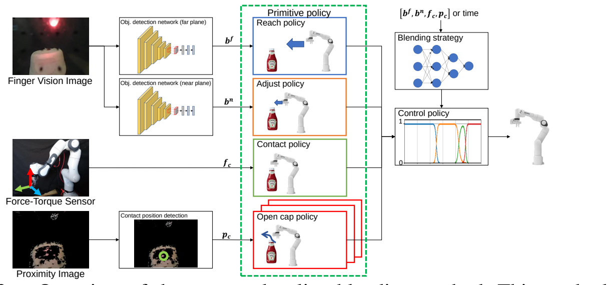

Fig. 2. Overview of the proposed policy blending method. This method consists of primitive policies, a blending strategy and a control policy. Each primitive policy uses a different modality and is learned individually.

to different tasks is limited to similar tasks, and additional learning is needed to add new tasks into the representations.

C. *Policy decomposition and planning*

Policy decomposition or blending is another approach to realize complex manipulation tasks without expansion of state space. [34] observes tactile array data for different grasp manipulation phases. Graph representations can describe complex manipulation tasks. [35] segments multiple different skills from human demonstration, and trains manipulation graphs which describe sequences of skill executions. [36] uses a graph representation to switch to recovery skills. A hierarchical approach is also proposed to solve skill planning problems [37]. [38] proposes different abstraction levels for primitive skills. Although training graphs or hierarchical representations can realize complex manipulation tasks, switching between different skills leads to discontinuities of motion, which can degrade task performance. Furthermore, adding new skills or generalizing to different tasks needs human design and additional training. [33] focuses on the composability of soft Q-learning algorithm and tries to learn new policy from decomposed policy for simple block stacking motion. [39] decomposes a neural network into task-specific and robot-specific modules. Although they propose a task agnostic control policy with policy decomposition, they fail to handle multimodal sensor signals.

III. POLICY BLENDING METHOD

RL is a promising approach for training multimodal manipulation tasks. However, the training process is not trivial and remains challenging. Our policy blending approach addresses existing problems of multimodal policy learning by manually dividing skills into separate parts and realizes a learning structure with low training cost and transfer of similar sequences of primitives to different tasks.

A. *Method overview*

The proposed policy blending method is based on the basic idea of manipulation phases. An entire manipulation motion has different phases, and each phase is seamlessly connected or mixed according to the task. Each phase has a different purpose and needs different sensor feedbacks. Therefore, each phase is linked to a modality, and various tasks are realized by sequencing together the primitive motion phases. Sequential primitives can then be blended together to create seamless transitions between phases, increasing robustness

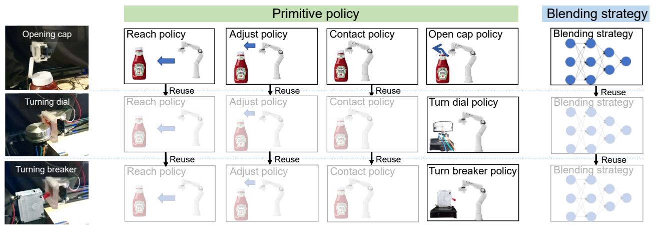

Fig. 3. Example of transfer to different tasks. The blending strategy and initial primitives can be reused for tasks that they were not trained on.

to sensor uncertainty during phase transitions and reducing the overall execution time. The overview of the proposed policy blending method is described in Fig.2. There are three components: the primitive policies, the blending strategy and the resulting control policy.

Each primitive policy corresponds to a manipulation phase, which is defined so that each policy utilizes a subset of modalities and is trained individually. Although we focus on using individual modalities, more than one modality can be used per primitive. It should be noted that segmenting the skill into phases is not the focus of this work and predefined primitive policies are used. The blending strategy calculates the blending ratio of each policy at each control time or state. It seamlessly connects primitive policies and creates continuous motions to perform target tasks. Since the blending strategy is not dependent on the primitive policies, their training can be separated. This blending approach seamlessly switches between primitives, allowing the robot to work towards multiple purposes in parallel and performs tasks in a fast and smooth fashion. The control policy is then the primitive policies combined according to the blending strategy and outputs the robot control command.

Decomposing skills into phases provides multiple advantages for training. Each phase only needs to learn feedback gains for a small set of sensor values, allowing the robot to combine modalities using the blending policy. The primitive policies can be performed sequentially, with one starting after the previous one ends. These primitive policies can thus be learned individually, with the blending providing performance and robustness increases. There are also other advantages for transfer to different tasks. Since primitive policies are modular and can be replaced, different tasks that require similar firm contacts can be performed by reusing the initial primitives and replacing the final primitive policy. Fig.3 shows example of task transfer. In this paper, three different tasks are explored: `opening cap`, `turning dial` and `turning breaker`. In order to perform the `opening cap` task, the `reach`, `adjust`, `contact` and `open cap` skill are trained. For the `turning dial` and `turning breaker` tasks, the `reach`, `adjust` and `contact` skills are reused, and only the `turn dial` and `turn breaker` skills are learned to replace the `open cap` skill respectively. The object detection networks are trained to detect the target object for each task. By reusing the primitive policies, the robot can learn to perform new tasks more quickly and focus on the novel aspects of the tasks.

Manipulation tasks in this paper are modeled as a finite-horizon Markov decision process (MDP), defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T}$: $\mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the state transition dynamics and $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function. The purpose of each RL problem is to find the optimal policy π which maximizes the expected sum of rewards $\mathbb{E}_\pi[\sum_t r(\boldsymbol{s_t}, \boldsymbol{a_t})]$, where the current state is $\boldsymbol{s_t} \in \mathcal{S}$ and action is $\boldsymbol{a_t} \in \mathcal{A}$. For each primitive policy and blending strategy a different reward functions r is defined according to the policy's purpose.

B. Policy representations

Dynamic motor primitives (DMPs) are used to represent each primitive policy. DMPs are differential equations which represent control policies with small number of parameters. The policy can be trained with various modality information to adopt the motion with different external environments. The advantage of using DMPs is that smooth and continuous movement is guaranteed even if modality information changes discontinuously. This paper uses the representation of DMPs based on [6]. This representation reformulates original DMP formulation [5] for generalization of skills with feature values. The primitive policy $\pi_w^p(\boldsymbol{s_t}, t)$ using formulation of DMPs is described as follows,

$$\pi_w^p(\boldsymbol{s_t}, t) = \ddot{y} = \alpha_z \left(\beta_z \tau^2 \left(y_0 - y\right) - \tau \dot{y}\right) + \tau^2 \sum_j \phi_j f(x; \boldsymbol{w_j}), \tag{1}$$

where y is the position of end-effector, y_0 is the initial position of end-effector, α_z and β_z are constants that define the spring and damper coefficients, x is the state of the canonical system, and f is a forcing function. τ is a temporal scaling factor which can be described as $\tau = 0.5/T$, where T is the standard duration time of the skill. The canonical state x monotonically decays from 1 towards 0 by the equation $\dot{x} = -\tau x$. The vector ϕ_j represents j^{th} feature values to amplify forcing function to adapt different external environments. The forcing function $f(x; \boldsymbol{w_j})$ is defined as

$$f(x; \boldsymbol{w_j}) = \alpha_z \beta_z \left(\frac{\sum_{k=1}^{K} \psi_k(x) w_{j,k} x}{\sum_{k=1}^{K} \psi_k(x)} + w_{i0} \psi_0(x) \right) \tag{2}$$

where the k^{th} element of the vector $\boldsymbol{w_j} \in \mathbb{R}^K$ is given by the weights $[\boldsymbol{w_j}]_k = w_{j,k}$, $\psi_k \forall k \in \{1, ..., K\}$ are Gaussian basis functions, and ψ_0 is a basis function that follows a minimum jerk trajectory from 0 to 1. The weights $\boldsymbol{w_j}$ in (1) are the parameters of primitive policy to be trained. In this paper, $\boldsymbol{w_j}$ is trained using RL.

C. Sensor modalities

As described in III-B, the feature vectors $\boldsymbol{\phi}$ play the role of adaptating to the external environment. This paper utilizes the feature vectors $\boldsymbol{\phi_j}$ as feedback terms using multiple sensor modalities' information. In this section, the details of each modality are described. We use 3 types of modalities: RGB images, contact force and proximity images.

Our proposed method assumes that RGB images are captured from two RGB cameras, one mounted on the end-effector and one at a fixed position in the environment. For the end effector camera we use a finger vision sensor, which includes a camera mounted under a transparent skin. RGB images are processed using a convolutional neural network (CNN) to detect target object position. To capture the target object, bounding box information (2D position, width and height) is calculated using Faster R-CNN [7]. The network uses a pre-trained ResNet [8] model with 152 layers, and fine-tunes it for the target objects. Since a camera is attached to the end-effector, the object size and features in the captured images change drastically depending on the camera's distance to the target object. Therefore, to acquire robust object detection results, two separate networks are trained to detect the objects when far away and nearby, which we refer to as *far plane* and *near plane* respectively. Bounding box information for far plane is denoted as $\boldsymbol{b^f} = [b^f_w, \boldsymbol{b^f_c}]$, where b^f_w is width and $\boldsymbol{b^f_c}$ is center position of bounding box in a image. For near plane, $\boldsymbol{b^n} = [b^n_w, \boldsymbol{b^n_c}]$. We use the width of the object to capture its visual scale as the width tended to be more reliable than the height. The bounding box provides an abstract representation of the target object, which makes it easier for simulation-to-reality (sim-to-real) transfer. The bounding box information from the two RGB cameras are combined to estimate the 3D position of the target object by epipolar triangulation. The continuously estimated 3D positions for the far and near plane networks are denoted as $\boldsymbol{g^f_{obj}}$ and $\boldsymbol{g^n_{obj}}$ respectively.

Contact forces are estimated from torque sensor at each joint of robot arm using a Jacobian transposed approach. The three axis force estimate $\boldsymbol{f_c}$ is utilized by the primitive policies for making and maintaining contact.

Proximity images are processed based on the RGB images from the end-effector camera. A small red flashing LED near the in-finger camera allows the robot to detect pixels of objects in close proximity to the finger, whose color values change with the LED. This image-based modality thus captures only the surface of the object that's near the camera. After processing the proximity images, the contact position is estimated by calculating center of mass of pixels in gray scale, which is denoted as $\boldsymbol{p_c}$.

D. Training primitive policies

In the following sections, we explain the feedback parameters of the different primitive policies and how the policies are learned. Every primitive policy is represented using DMPs described in III-B.

`reach`, `adjust` skills: The purpose of these two skills is to move the robot toward the target object. The `reach` skill moves the robot towards the object and the `adjust` skill finely adjusts the end-effector position before making contact. These skills consist of nominal and feedback motions. The nominal motion defines the basic motion, which is independent of sensor feedback, and feedback motion adjusts the nominal motion according to sensor feedback.

Although the nominal motion defines sensor independent motion, it should be robust to variations in the robot's initial position and the target object position. The DMP equations (1) are able to adjust the motion according to the initial robot and object positions by using the amplitude term ϕ. Having a difference between target object position and initial end-effector (EE) position as an amplitude $\boldsymbol{\phi_1} = \boldsymbol{d^f} = \boldsymbol{g^f_{obj}} - \boldsymbol{y_{0EE}}$ or $\boldsymbol{\phi_1} = \boldsymbol{d^n} = \boldsymbol{g^n_{obj}} - \boldsymbol{y_{0EE}}$, the DMPs create a motion that adjusts to different initial positions. The superscript: $\boldsymbol{f}$ represents "far plane" for `reach` skill, and $\boldsymbol{n}$ represents "near plane" for `adjust` skill. $\boldsymbol{g_{obj}}$ is updated over time.

However, the goal positions of `reach` and `adjust` skills are not equivalent to the target object positions. To offset the goal position of the DMP motions for suitable contact placement, goal offset values should also be learned. We model the goal offsets by including fixed amplitude values $\boldsymbol{\phi_2} = \boldsymbol{1}$, and allow the robot to learn the corresponding w_i weights as goal offset values.

For additional feedback, bounding box information is utilized to adjust the robot motion so that robot moves toward the target object. A goal position of bounding box $\boldsymbol{b^f_g}$, $\boldsymbol{b^n_g}$ is extracted from demonstrations, and the difference between $\boldsymbol{b^f_g}$, $\boldsymbol{b^n_g}$ and the current center position of bounding box $\boldsymbol{b^f_c}$, $\boldsymbol{b^n_c}$ is passed as the third amplitude value $\boldsymbol{\phi_3} = \boldsymbol{b^f_g} - \boldsymbol{b^f_c}$ or $\boldsymbol{\phi_3} = \boldsymbol{b^n_g} - \boldsymbol{b^n_c}$.

`contact` skill: The purpose of this skill is to make contact with the target object and then keep an appropriate contact force. Since the nominal motion is independent from any sensor values, a fixed amplitude vector $\boldsymbol{\phi_1} = \boldsymbol{1}$ is used. The feedback motion is trained with the contact force feedback. The difference between the target contact force extracted from demonstrations $\boldsymbol{f_g} \in \mathbb{R}^3$ and the current force $\boldsymbol{f_c} \in \mathbb{R}^3$ give the second amplitude $\boldsymbol{\phi_2} = \boldsymbol{f_g} - \boldsymbol{f_c}$.

`open cap`, `turn dial`, `turn breaker` skill: These skills perform manipulation tasks once contact with the target objects has been established. The nominal motion is trained by using a fixed amplitude vector $\boldsymbol{\phi_1} = \boldsymbol{1}$. The feedback motion is trained with the contact position feedback calculated from the proximity images. The difference between the target contact position extracted from demonstrations $\boldsymbol{p_g} \in \mathbb{R}^2$ and the current contact position $\boldsymbol{p_c} \in \mathbb{R}^2$ gives the second amplitude vector $\boldsymbol{\phi_2} = \boldsymbol{p_g} - \boldsymbol{p_c}$.

The primitive policies are trained by optimizing parameters $\boldsymbol{w}$ in (1) with feature vectors ϕ. Optimization of parameters takes two steps. First, all DMP parameters are initialized by imitation learning. A human demonstrates each primitive policy with gravity compensation control mode, and 10 trajectories are recorded. By solving linear regression problem from (1) and (2), the policy parameters $\boldsymbol{w}$ are initialized. Afterwards, RL is used to optimize the policy parameters through trial and error. In particular, we utilize the relative entropy policy search (REPS) algorithm [3]. The advantage of REPS is that reward maximization process is quite light and the loss of information is bounded when a policy is updated, which leads to faster and better convergence.

E. Training blending strategy

Each primitive policy is blended into a control policy that then defines the desired trajectory acceleration. Since each primitive policy has a different purpose and generates different motions, they should be executed at appropriate timings and blended according to the current state or elapsed time. The blending strategy is a policy which calculates a blending ratio of each primitive policy at a certain control time or environment state. The control policy $\pi^c_{w'}$ is defined by the blending strategy as follows,

$$\pi^c_{w'} = \sum_{m=1}^{M} [\pi^b_\theta(\boldsymbol{s_t})]_m [\hat{\boldsymbol{\pi}}^{\boldsymbol{p}}_{\boldsymbol{w}}]_m \tag{3}$$

where $\pi^b_\theta(\boldsymbol{s_t})$ is the blending strategy, $\hat{\boldsymbol{\pi}}^{\boldsymbol{p}}_{\boldsymbol{w}} = [\pi^{p1}_w(\boldsymbol{s_t}, t), \pi^{p2}_w(\boldsymbol{s_t}, t), \cdots, \pi^{pM}_w(\boldsymbol{s_t}, t)]$ is a vector of primitive policies, and M is the number of primitive policies. For RL of the blending strategy, action $\boldsymbol{a_t}$ denotes the blending ratio of each primitive policy at time t, which can be denoted as $\boldsymbol{a_t} = \pi^b_\theta(\boldsymbol{s_t})$. This paper proposes two types of representation of blending strategies: time dependent and state dependent.

Time Dependent Blending Strategy: The time dependent case calculates blending ratios according to the elapsed time.

$$\pi^b_\theta(\boldsymbol{s_t}, t) = \text{softmax}(\boldsymbol{w^m_\theta}), \tag{4}$$

where $\boldsymbol{w^m_\theta}$ is the time dependent weight for the m-th skill which can be calculated as $\boldsymbol{w^m_\theta} = \sum_{l=1}^{L} \psi_l(t)[\boldsymbol{\theta}]^m_l$, where $\psi_l(t)$ is the l-th Gaussian basis function and $[\boldsymbol{\theta}]^m_l$ is the policy parameters to be trained for l-th Gaussian basis function of primitive policy m. Other functions could be used to capture more complex blending behaviours, but more flexible and detailed representations will often also require more data. The learning process is based on episodic RL, where the trajectory $\boldsymbol{\tau}$ can be represented as $\boldsymbol{\tau} = \{\boldsymbol{s_0}, \boldsymbol{a_0}, \boldsymbol{s_1}, \boldsymbol{a_1}, ...\}$, and the robot will receive an accumulated reward $R(\boldsymbol{\tau})$ for the entire trajectory. The purpose of RL for time dependent case is to maximizing the task's expected accumulated reward $J_{\pi^b_\theta} = \mathbb{E}_{\pi^b_\theta}[R(\boldsymbol{\tau})]$. Trajectories are recorded by running control policy in (3). The parameters $\boldsymbol{\theta}$ are then again trained with Relative Entropy Policy Search algorithm (REPS) [3].

State Dependent Blending Strategy: The state dependent case calculates blending ratios according to the current sensor information. The policy is represented as a neural network which has two hidden layers of size 256 and a soft-max layer as the output,

$$\pi^b_\theta(\boldsymbol{s_t}) = f(\boldsymbol{s_t}, \boldsymbol{\theta}). \tag{5}$$

where $\boldsymbol{\theta}$ is a set of parameters for the state dependent blending strategy and state $\boldsymbol{s_t}$ is the sensor information at time step t. The robot will receive the reward $r(\boldsymbol{s_t}, \boldsymbol{a_t})$ when executing blending ratio $\boldsymbol{a_t}$ with state $\boldsymbol{s_t}$. The state is represented as a concatenated vector of sensor information: $\boldsymbol{s_t} = [\boldsymbol{b^f}, \boldsymbol{b^n}, \boldsymbol{f_c}, \boldsymbol{p_c}]$, The goal of the RL for state dependent blending strategy is to maximize the expected reward $J_{\pi^b_\theta} = \mathbb{E}_{\pi^b_\theta}[\sum_t \gamma r(\boldsymbol{s_t}, \boldsymbol{a_t})]$, where $\gamma \in (0, 1]$ is the discount factor. The parameters $\boldsymbol{\theta}$ are trained with Soft Actor Critic Algorithm (SAC) [4].

Each primitive policy is represented by a DMP, which is a time dependent policy. On the other hand, the blending strategy can be either time dependent or state dependent. Therefore, an inconsistency between time and state dependency exists in (3). Furthermore, the DMP equation in (1) has initial position y_0, which are varied between primitive skills and dependent on blending policy. In order to solve the dependency problems in (3), the canonical system x, which is the time dependent term in the DMP equation, and initial position y_0 are updated according to the blending strategy. Let $\boldsymbol{x}(k) = \{\boldsymbol{x^1}(k), \boldsymbol{x^2}(k), ..., \boldsymbol{x^M}(k)\}^T$ be a vector of canonical systems of all M primitive policies at time step k, $\boldsymbol{Y_0}(k) = \{\boldsymbol{y^1_0}(k), \boldsymbol{y^2_0}(k), ..., \boldsymbol{y^M_0}(k)\}^T$, $\boldsymbol{Y_0}(k) \in \mathbb{R}^{M \times N}$ be a vector of N-dimensional initial end-effector positions of all primitive policies at time step k and $\boldsymbol{y}(k)$ be the current position, then $\boldsymbol{x}$ and $\boldsymbol{Y_0}$ are updated as follows,

$$\boldsymbol{x}(t+1) = \boldsymbol{x}(t) + \pi^b_\theta(\boldsymbol{s_t}) \otimes \Delta\boldsymbol{x} \tag{6}$$

$$\boldsymbol{Y_0}(t+1) = \frac{\pi^b_\theta(\boldsymbol{s_t}) \otimes \boldsymbol{Y_0}(t)^T + \Delta\pi^b_\theta(\boldsymbol{s_t})\boldsymbol{y}(t)^T}{\pi^b_\theta(\boldsymbol{s_{t+1}})}, \tag{7}$$

where $\boldsymbol{\Delta x} = -\tau \boldsymbol{x}(t)\Delta t$, $\Delta\pi^b_\theta(\boldsymbol{s_t}) = \pi^b_\theta(\boldsymbol{s_{t+1}}) - \pi^b_\theta(\boldsymbol{s_t})$ and $\otimes$ represents element-wise multiplication. The intuitive explanation of this operation is that when one primitive policy starts running, the canonical system of this primitive policy is updated to generate motions of this policy and set current position as the initial position of this primitive policy.

IV. EXPERIMENTS: SETUP

The primary goal of our experiments is to evaluate the effectiveness of the proposed policy blending method. In particular, evaluations are conducted to answer the following questions: 1) Does multimodal sensing make manipulation movements more precise and reliable? 2) Can the proposed blending strategy be trained to blend different skills to perform the task successfully and produce a smooth trajectory? 3) Does the proposed method scale to different tasks? 4) What sensor features are crucial for contact-rich tasks?

Task Setup: Three different tasks were prepared for evaluation: `opening cap`, `turning dial` and `turning breaker` as shown in Fig.1. The `opening cap` task is the task of opening the ketchup bottle by moving the robot finger in from the side and then up and subsequently keeping the cap angle greater than 90 degrees. `turning dial` is the task of rotating the dial switch by moving the robot finger horizontally and turning the switch by an angle greater than 60 degrees. `turning breaker` involves turning a breaker switch by moving the robot finger from down to up and keep the breaker knob topside.

Experiment Setup: A Franka Emika panda, a 7-DoF torque-controlled arm, is used for both simulation and real experiments. For simulation, NVIDIA® Isaac gym, which can run multiple parallel simulations on the GPU [9], is selected. As described in III-C, two RGB cameras are placed on the end-effector and a fixed position in the environment. In the real environment, a finger vision tactile sensor [2] is used as the in-hand camera and an Azure Kinect camera is used as the fixed camera. Finger vision consists of an RGB camera and a transparent silicone rubber which can observe both the external scene through transparent silicone rubber and contact area on the rubber surface. Contact force at the end-effector is estimated from the torque sensor at each joint of Franka Emika panda arm. Proximity image is created from RGB image captured by finger vision. For simulation, the proximity image is created by changing the rendering distance to show only near objects within 1cm. For real environment, a red blinking LED is attached to the finger vision (Fig.1), and only the environment near the object is detected using reflected red LED light using color threshold.

For simulation environment, RGB images of finger vision and fixed camera are captured at 33Hz, and contact force is sampled at 100Hz. In the real environment, RGB images are captured at 5Hz and contact force is sampled at 30Hz. All RGB images are down-sampled to 320 x 240 and input to Faster R-CNN network.

Rewards are calculated according to the angle of the cap, dial and breaker. For the real environment, different types of sensors are attached to the target object to calculate rewards. A flex bend sensor estimates the angle of the ketchup bottle cap, a potentiometer directly measures the angle of the dial knob and a voltage indicates the breaker on/off status.

In each evaluation, performance is evaluated from 100 samples in the simulation or 25 samples in real environment. Action $\boldsymbol{a_t} \in \mathbb{R}^3$ is the desired acceleration of the end-effector in Cartesian space. The orientation of end-effector and the position of the target objects are fixed. The initial position of end-effector is uniformly sampled within a 20cm cubic area.

V. EXPERIMENT RESULTS

A. *Sensory feedback for individual primitives*

In the first experiment, we evaluate the effectiveness of tactile and proximity sensing for adapting primitives. Each primitive policy is executed, and the improvement of accuracy between with and without sensor values cases are compared. For `reach` and `adjust` skills, 3D position errors between the target position and the end-effector position is observed. For the `contact` skill, the contact force error between target and contact force at the termination time is evaluated. Since accuracy of contact position is crucial for each task, the accuracy of contact position after running `reach`, `adjust` and `contact` skill is also evaluated. This evaluation is conducted for the `opening cap` task in simulation environment. Table I describes the results. Table I indicates that sensor feedback plays an important role to improve accuracy of each skill. The contact position error after three skills are executed is also improved with sensor feedback. Without sensor feedback, the error is larger than 1.5cm, which is crucial when a thin robot finger is used.

B. *Blending strategy*

The effectiveness of both time dependent and state dependent blending strategies are evaluated and compared with two baseline methods. One of the baseline method: "Baseline 1" sequentially executes each skill in the fixed order. Each skill has fixed duration and is executed one after another. This baseline represents the common approach to executing sequences of skills. The second baseline method: "Baseline 2" terminates the skill at a trigger timing, and executes the next skill without waiting for completion of the skill. The order of skill executions is fixed and the trigger timings for each skill are learned to maximize the success rate using REPS algorithm. The proposed blending strategies and baseline methods are evaluated in terms of success rate, average max acceleration and manipulation duration as shown in Table II. The proposed blending strategy methods show higher success rates and lower accelerations than Baseline 2. Baseline 1 still shows high success rate; however, its trajectory is not smooth. Comparison between the time and state dependent blending strategies shows that the y-z direction accelerations are higher in the state dependent case. The best explanation to this result is that state dependent case is more sensitive to bounding box information, which causes higher acceleration than for the time dependent case. The results also show that the blending method makes manipulation duration shorter, especially with the state dependent case.

TABLE I
POSITION ERROR AND CONTACT FORCE ERROR OF EACH SKILL

Skill	Without sensor	With sensor
`reach` (mm)	41.6 (±33.1)	24.2 (±9.3)
`adjust` (mm)	23.1 (±9.7)	8.90 (±4.2)
`contact` (N)	8.33 (±8.92)	1.44 (±0.34)
`reach+adjust+contact`(mm)	15.7 (±36.0)	5.70 (±4.2)

TABLE II
COMPARISON BETWEEN BASELINE METHOD AND PROPOSED METHOD

Method	Success rate(%)	Avg. max acceleration (m/s^2) x	y	z	Duration(s)
Baseline 1	97	7.34 (±1.3)	1.40 (±0.3)	2.37 (±0.4)	16.0
Baseline 2	85	14.7 (±2.5)	4.68 (±0.7)	3.79 (±0.9)	14.8
Blending (time)	97	3.01 (±1.0)	0.23 (±0.3)	0.26 (±0.1)	12.0
Blending (state)	95	1.04 (±0.2)	1.18 (±0.2)	0.32 (±0.3)	8.7

Fig.4 shows the relationship between the sensors and trained blending strategy for the cap opening task in time dependent and state dependent cases. For both cases, primitive policies are blended according to the sensor events. For example, switching between the `reach`, `adjust` and `contact` skills occurs when bounding box position centers to the image and switching between `contact` and `open cap` skill occurs when contact force is detected. The skill switching time is more accurate in the state dependent case. Since each modality has different operating ranges, sensor values are noisy and inaccurate outside of their operating range. This usually makes it difficult to learn control policies that use all sensor modalities all the time. Even under such difficult cases, a blending strategy can be successfully trained to perform the cap opening task. Fig.5 shows a comparison of the training curve between no blending case, baseline methods and blending method. In no blending case, a whole robot trajectory is trained with all sensor values with the REPS algorithm. The results shows that the proposed blending method drastically reduces learning cost and robot motion for the `opening cap` task, and it can be trained within a practical number of episodes.

C. *Transfer to different tasks*

The goal of the skill modularity is realizing a control policy which can create different motions for various task purposes

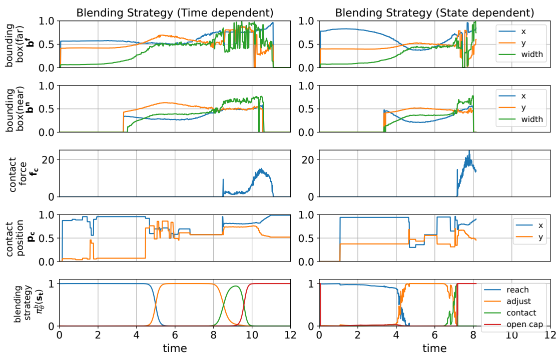

Fig. 4. Sensor values and trained blending strategy. Each sensor value and blending ratio calculated by the blending strategy are shown. Left side is for time-dependent case and right side is for state-dependent case.

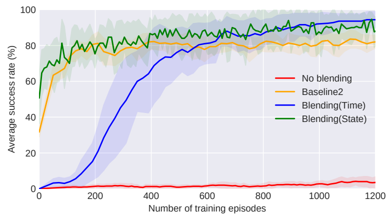

Fig. 5. Success rates during RL training for `opening cap` task. The results are compared with two baseline methods and no blending method.

without significant additional training. In this evaluation, we investigate the transfer of the skills to different manipulation tasks that require firm contact with the object. The primitive skills: `reach`, `adjust`, `contact`, `open cap` and blending strategies are trained for the `opening cap` task, while `turn dial` and `turn breaker` skill are trained for `turning dial` and `turning breaker` task respectively. Each task thus has a predefined different subset of primitive policies which includes some policies from other tasks (Fig.3). Blending strategies are reused for tasks without additional training. This evaluation is conducted for both simulation and real environment, and the success rates of three tasks are compared. For the real environment, every policy is directly transferred from simulation to the real world without any additional training.

Fig.6 describes the comparison of success rates between different tasks. The proposed blending strategy shows high success rates for different tasks in simulation. The results also indicate that success rate is still high for real executions. The good sim-to-real performance is due to the fact that the blending strategy is not greatly affected by environment differences and blending skills guarantee continuity of trajectory and switching modality. By contrast, the sim-to-real performance is worse for the Baseline 2 case which switches skills discontinuously. The time-dependent blending strategy shows higher success rate than the state-dependent as it is less susceptible to sensor measurement noise. However, the generalization ability of time-dependent case is limited for only temporally similar sequences. For example, the time-dependent case will not generalize well if the target object moves far away or if a human push back the robot. By contrast, state-dependent case can switch back to a previous primitive policy. The DMPs help to create temporally similar sequences, thus making the time-dependent blending viable for different starting positions. The success rate of the turn

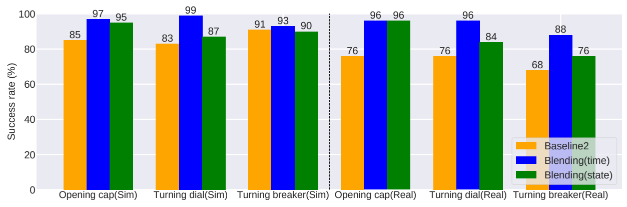

Fig. 6. Success rates of three different tasks. Success rate is compared with different blending methods and tasks for both simulation and real environment. Left half is the results of simulation and right half is the results of real environment.

TABLE III

RESULTS OF ABLATION STUDY

	Modality						Success rate	
	d^f	d^n	b^f	b^n	f_c	p_c	Sim	Real
1	✓	✓	✓	✓	✓	✓	82%	80%
2	✓	✓	–	–	✓	✓	78%	76%
3	–	–	✓	✓	✓	✓	**95%**	**96%**
4	–	–	✓	✓	✓	–	80%	36%
5	–	–	✓	✓	–	✓	82%	44%
6	–	–	✓	✓	–	–	74%	24%

breaker task in the real environment is lower than the success rates of other tasks. This is because the size of the breaker is much smaller than the ketchup bottle cap and dial knob. This causes object detection failures and mismatch of sensor values for blending strategy. A scaling factor may be added to compensate for the smaller size of the object.

Transfer to different tasks works well for similar firm contacts where the number of primitive policies is the same and the structure of the sequence is similar with only the last skill being altered. On the other hand, complicated tasks which take various number of primitive policies and reactive switch of the tasks will usually require additional training. An additional upper policy or a hierarchical structure may be needed so that sequences can be executed reactively by choosing primitive policies or blending strategies depending on the manipulation task.

D. *Ablation study and sim-to-real*

In this section, an ablation study for the state dependent blending strategy is conducted to investigate which particular modalities are important for high performance. At the same time, sim-to-real results are also observed to investigate which modality are important for sim-to-real. Total 6 ablation patterns were evaluated. The sensor values includes the same modality information which is used for training the primitive policies as described in III-D. The results are shown in Table III. The success rate in simulation and real environment are shown for each ablation pattern. Patterns 1, 2 and 3 show the importance of distance and bounding box information. Success rate deteriorates without a bounding box, and this shows that bounding box information is critical for performance. On the other hand, distance information deteriorates the performance. This is an interesting finding and can be explained by how the policy network overfits to distance information and thus loses robustness against position error. Patterns 3,4,5 and 6 show the importance of each sensor value. Pattern 3 shows highest performance for both sim and real environment and disabling force f_c or proximity p_c or both leads poor performance, especially for real environment. This can be explained by how the bounding box information is noisy in real environment and $\boldsymbol{f_c}$ or $\boldsymbol{p_c}$ information play more important role to switch the primitive policies reliably.

VI. CONCLUSIONS

We present policy blending method for multimodal manipulation. In the proposed method, the robot control policy is divided into separate primitive policies so that each policy uses a subset of primitive modalities, and the primitive policies are subsequently blended by a learned blending strategy. Finger vision and a fore-torque sensor are used to acquire multimodal information which gives the robot tactile and proximity feeling. Our experiments show that tactile and proximity information improve the performance of manipulations. Also, the proposed method enables training of control policies within practical times even if each sensor has a different operating range, dynamic range and precision. The trained policies for establishing contact can be transferred to different tasks by replacing only the final primitive, which is effective for practical use. The evaluation results show high success rates for three different tasks, and trained policies could be transferred to real environment without additional training. For future work, the proposed method can be extended to other manipulation tasks which require high DoF motion including orientation. Also, a hierarchical structure is a possible solution to handle complicated tasks which require prioritized modality sensing.

REFERENCES

[1] R. S. Johansson, J. R. Flanagan, "Coding and use of tactile signals from the fingertips in object manipulation tasks," in *Nat. Rev. Neurosci.* 10, 345–359 (2009).

[2] A. Yamaguchi and C. G. Atkeson, "Combining finger vision and optical tactile sensing: Reducing and handling errors while cutting vegetables," in *Humanoids*, 2016, pp. 1045-1051.

[3] J. Peters, K. Mulling, and Y. Altun, "Relative entropy policy search," In AAAI., 2010.

[4] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," in *ICML*, 2018, 80:1861-1870.

[5] A. J. Ijspeert, J. Nakanishi and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2002, pp. 1398-1403 vol.2.

[6] O Kroemer, and G. S. Sukhatme, "Learning Relevant Features for Manipulation Skills using Meta-Level Priors," Kroemer-2016-112245, Carnegie Mellon University, Pittsburgh, PA, May, 2016.

[7] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137-1149, 2017.

[8] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in *CVPR.*, 2016, pp. 770-778.

[9] J. Liang, V. Makoviychuk, A. Handa, N. Chentanez, M. Macklin and D. Fox, "GPU-Accelerated Robotic Simulation for Distributed Reinforcement Learning," in *Proc. Conf. Robot Learn.*, 2018

[10] P. K. Allen, A. Miller, P. Oh and B. Leibowitz, "Integration of Vision, Force and Tactile Sensing for Grasping,", in *Int. J. Intell. Mach.*, 4.

[11] J. S. Son, R. Howe, J. Wang and G. D. Hager, "Preliminary results on grasping with vision and touch," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 1996, pp. 1068-1075 vol.3.

[12] M. Prats, P. J. Sanz and A. P. del Pobil, "Vision-tactile-force integration and robot physical interaction," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2009, pp. 3975-3980.

[13] H. Bae, S. Jeon and J. P. Huissoon, "Multimodal sensory integration for estimating rigid body motion of a dynamically manipulated object," American Control Conference, 2014, pp. 3437-3442.

[14] Q. Li, R. Haschke and H. Ritter, "A visuo-tactile control framework for manipulation and exploration of unknown objects," in *Humanoids*, 2015, pp. 610-615.

[15] J. Bimbo, L. D. Seneviratne, K. Althoefer and H. Liu, "Combining touch and vision for the estimation of an object's pose during manipulation," in *Proc. IEEE/RSJ IROS*, 2013, pp. 4021-4026.

[16] J. Bimbo et al., "Object pose estimation and tracking by fusing visual and tactile information," in *IEEE MFI*, 2012, pp. 65-70.

[17] C. Yang and N. F. Lepora, "Object exploration using vision and active touch," in *Proc. IEEE/RSJ IROS*, 2017, pp. 6363-6370.

[18] V. Müller and N. Elkmann, "Multimodal Bin Picking System with Compliant Tactile Sensor Arrays for Flexible Part Handling*," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 2824-2830.

[19] J. Sung, I. Lenz and A. Saxena, "Deep multimodal embedding: Manipulating novel objects with point-clouds, language and trajectories," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 2794-2801.

[20] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal Deep Learning," in *Proc. ICML*, 2011, pp. 689–696.

[21] M. A. Lee et al., "Making Sense of Vision and Touch: Learning Multimodal Representations for Contact-Rich Tasks," in *IEEE Trans. Robot.*, vol. 36, no. 3, pp. 582-596, June 2020.

[22] K. Noda, H. Arie, Y. Suga and T. Ogata, "Multimodal integration learning of object manipulation behaviors using deep neural networks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 1728-1733.

[23] F. Li, Q. Jiang, W. Quan, S. Cai, R. Song and Y. Li, "Manipulation Skill Acquisition for Robotic Assembly Based on Multi-Modal Information Description," in IEEE Access, vol. 8, pp. 6282-6294, 2020.

[24] R. Calandra et al., "More Than a Feeling: Learning to Grasp and Regrasp Using Vision and Touch," in *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3300-3307, Oct. 2018.

[25] S. Hasegawa, K. Wada, S. Kitagawa, Y. Uchimi, K. Okada and M. Inaba, "GraspFusion: Realizing Complex Motion by Learning and Fusing Grasp Modalities with Instance Segmentation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 7235-7241.

[26] T. Bhattacharjee, H. M. Clever, J. Wade and C. C. Kemp, "Multimodal Tactile Perception of Objects in a Real Home," in *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 2523-2530, July 2018.

[27] U. Großekathöfer, A. Barchunova, R. Haschke, T. Hermann, M. Franzius and H. Ritter, "Learning of object manipulation operations from continuous multimodal input," in *Humanoids*, 2011, pp. 507-512.

[28] D. Park, Z. Erickson, T. Bhattacharjee and C. C. Kemp, "Multimodal execution monitoring for anomaly detection during robot manipulation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 407-414.

[29] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni and S. Levine, "Vision-Based Multi-Task Manipulation for Inexpensive Robots Using End-to-End Learning from Demonstration," in *Proc. IEEE Int. Conf. Robot. Autom.*, Brisbane, QLD, 2018, pp. 3758-3765.

[30] O. Mees, A. Eitel and W. Burgard, "Choosing smartly: Adaptive multimodal fusion for object detection in changing environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 151-156.

[31] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," in *JMLR*, vol. 17, no. 1, 2016.

[32] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 3406-3413.

[33] T. Haarnoja, V, Pong, A. Zhou, M. Dalal, P. Abbeel, S. Levine, "Composable Deep Reinforcement Learning for Robotic Manipulation," in *Proc. IEEE Int. Conf. Robot. Autom.*. 2018. pp. 6244-6251.

[34] N. Hendrich, D. Klimentjew and J. Zhang, "Multi-sensor based segmentation of human manipulation tasks," in *IEEE MFI*, 2010.

[35] Z. Su, O. Kroemer, G. E. Loeb, G. S. Sukhatme and S. Schaal, "Learning Manipulation Graphs from Demonstrations Using Multimodal Sensory Signals," in *Proc. IEEE ICRA*, 2018, pp. 2758-2765.

[36] A. S. Wang and O. Kroemer, "Learning Robust Manipulation Strategies with Multimodal State Transition Models and Recovery Heuristics," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 1309-1315.

[37] J. Barry, L. P. Kaelbling and T. Lozano-Pérez, "A hierarchical approach to manipulation with diverse actions," in *ICRA.*, 2013, pp. 1799-1806.

[38] M. weser and J. Zhang, "Autonomous planning for mobile manipulation services based on multi-level robot skills," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2009, pp. 1999-2004.

[39] C. Devin, A. Gupta, T. Darrell, P. Abbeel and S. Levine, "Learning modular neural network policies for multi-task and multi-robot transfer," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 2169-2176.