Learning Sensorimotor Primitives of Sequential Manipulation Tasks from Visual Demonstrations

Junchi Liang, Bowen Wen, Kostas Bekris and Abdeslam Boularias

Abstract— **This work aims to learn how to perform complex robot manipulation tasks that are composed of several, consecutively executed low-level sub-tasks, given as input a few visual demonstrations of the tasks performed by a person. The sub-tasks consist of moving the robot's end-effector until it reaches a sub-goal region in the task space, performing an action, and triggering the next sub-task when a pre-condition is met. Most prior work in this domain has been concerned with learning only low-level tasks, such as hitting a ball or reaching an object and grasping it. This paper describes a new neural network-based framework for learning simultaneously low-level policies as well as high-level policies, such as deciding which object to pick next or where to place it relative to other objects in the scene. A key feature of the proposed approach is that the policies are learned directly from raw videos of task demonstrations, without any manual annotation or post-processing of the data. Empirical results on object manipulation tasks with a robotic arm show that the proposed network can efficiently learn from real visual demonstrations to perform the tasks, and outperforms popular imitation learning algorithms.**

I. Introduction

Complex manipulation tasks are performed by combining low-level sensorimotor primitives, such as grasping, pushing and simple arm movements, with high-level reasoning skills, such as deciding which object to grasp next and where to place it. While low-level sensorimotor primitives have been extensively studied in robotics, learning how to perform high-level task planning is relatively less explored.

High-level reasoning consists of appropriately chaining low-level skills, such as picking and placing. It determines when the goal of a low-level skill has been reached, and the pre-conditions for switching to the next skill are satisfied. This work proposes a unified framework for learning both low and high level skills in an end-to-end manner from visual demonstrations of tasks performed by people. The focus is on tasks that require manipulating several objects in a sequence. Examples include stacking objects to form a structure, as in Fig. 1, removing lug nuts from a tire to replace it, and dipping a brush into a bucket before pressing it on a surface for painting. These tasks are considered in the experimental section of this work. For all of these tasks, the pre-conditions of low-level skills depend on the types of objects as well as their spatial poses relative to each other, in addition to the history of executed actions. To support the networks responsible for the control policies, this work uses a separate vision neural network to recognize the objects and to track their 6D poses both over the demonstration videos as well as during execution. The output of the vision network is the semantic category of each object and its 6D pose relative to other objects. This output along with the history of executed actions is passed to a high-level reasoning neural network, which selects a pair of two objects that an intermediate level policy needs to focus its *attention* on.

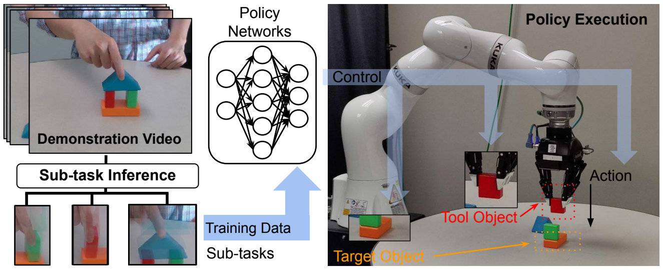

Fig. 1: System overview: (top left) Video demonstrations of sequential manipulation tasks performed by a person. (bottom left) The manipulated objects are tracked in the input video to automatically identify sub-tasks. (center) Given the tracking, policy networks are trained to perform high-level reasoning and compute low-level controls. (right) The output of the policy networks is forwarded to a robot, which manipulates the same objects as in the demonstrations without other manual annotation. It is important to note that the objects are always **randomly** placed on the table at the beginning of each demonstration and each test scenario.

The first object is referred to as *the tool*, and to the second one as *the target*. In a stacking task, the tool is the object grasped by the robot and the target is a table or any other object on top of which the tool will be placed. In the painting task, the tool is the brush and the target is the paint bucket or the canvas. If no object is grasped, then the tool is the robot's end-effector and the target is the next object that needs to be grasped or manipulated. An intermediate-level network receives the pair of objects indicated by the high-level reasoning network, their 6D poses relative to each other, and a history of executed actions. The intermediate-level network returns a sub-goal state, defined as a *way-point* in $SE(3)$. Finally, a low-level neural network generates the end-effector's motion to reach the way-point. The policy neural networks are summarized in Fig. 2.

While the proposed formulation is not exhaustive, it allows to cast a large range of manipulation tasks, and to use the same network to learn them. The proposed architecture requires only raw RGB-D videos, without the need to segment them into sub-tasks, or even to indicate the number of sub-tasks. The efficacy of the method is demonstrated in extensive experiments using real objects in visual demonstrations, as well as both simulation and a real robot for execution.

This work was supported by NSF awards IIS 1734492, IIS 1846043 and and IIS 2132972. The authors are with the Department of Computer Science, Rutgers University, NJ, USA. Emails: {junchi.liang, bowen.wen, kostas.bekris, abdeslam.boularias}@rutgers.edu.

II. Related Work

Most of the existing techniques in imitation learning in robotics are related to learning basic low-level sensorimotor primitives, such as grasping, pushing and simple arm movements [1], [2]. The problem of learning spatial preconditions of manipulation skills has been addressed in some prior works [3], [4]. Random forests were used [3] to classify configurations of pairs of objects, obtained from real images, into a family of low level skills. However, the method presented in [3] considers only static images where the objects are in the vicinity of each other [3], in contrast to the proposed model, which continuously predicts low-level skills while the objects are being manipulated and moved by the robot. Moreover, it does not consider complex tasks that are composed of several low-level motor primitives [3].

A closely related line of work models each sub-task as a funnel in the state space, wherein the input and output regions of each sub-task are modeled as a multi-modal Gaussian mixture [4], [5], and learned from explanatory data through an elaborate clustering process. Explicit segmentation and clustering have also been used [6]. Compared to these methods, the proposed approach is simple to reproduce and uses significantly less hyper-parameters since it does not involve any clustering process. Our approach trains an LSTM to select and remember pertinent past actions. The proposed approach also aims for data-efficiency through an attention mechanism provided by the high-level network. Hierarchical imitation learning with high and low level policies is investigated in recent work [7], [8]. These methods require ground-truth labeling of each sub-task to train the high-level policy, while the proposed method is unsupervised.

Skill chaining was considered in other domains, such as 2D robot navigation [9]. Long-horizon manipulation tasks have also been solved by using symbolic representations via Task and Motion Planning (TAMP) [10], [11], [12], [13]. Nevertheless, all the variables of the reward function in these works are assumed to be known and fully observable, in contrast to the proposed approach. A finite-state machine that supports the specification of reward functions was presented and used to accelerate reinforcement learning of structured policies [14]. In contrast to the proposed method, the structure of the reward machine was assumed to be known. A similar idea has also been adopted in other efforts [15], [16].

While 6D poses and labels of objects are provided from a vision module [17] in the proposed approach, other recent works have shown that complex tasks can be completed by learning directly from pixels [18], [19], [20], [21], [22], [23], [24]. This objective is typically accomplished by using compositional policy structures that are learned by imitation [18], [19], or that are manually specified [20], [21]. Some of these methods have been used for simulated control tasks [25], [26], [27]. These promising end-to-end techniques still require orders of magnitude more training trajectories compared to methods like the one proposed here, which separates the object tracking and policy learning problems.

III. Problem Formulation and Architecture

This approach employs a hierarchical neural network for learning to perform tasks that require consecutive manipulation of multiple objects. The assumption is that each scene contains at most n objects from a predefined set $\mathcal{O} = \{o^1, o^2, \ldots, o^n\}$. The robot's end-effector is included as a special object in $\mathcal{O}$. The robot receives as inputs at each time-step t sensory data as an observation $z_t = (e_t, \langle l_t^1, \ldots, l_t^m \rangle, p_t)$, where $e_t \in \mathbb{R}^3 \times \mathbb{SO}(3)$ is the 6D pose of the end-effector in the world frame, m is the maximum number of objects present in the scene, l_t^i is the semantic label of object o^i, and p_t is a $7n \times 7n$ matrix that contains the 6D poses of all objects relative to each other, i.e., $p_t[o^i, o^j]$ is a 7-dim. vector that represents o^i's orientation and translation in the frame of object o^j. The objects have known geometric models and have fixed frames of reference defined by their centers and 3 principal axes. The objects are detected and tracked using the technique presented in Section V-B. The maximum number of objects n is fixed a priori.

The system returns at each time-step t an action $a_t \in \mathbb{R}^3 \times \mathbb{SO}(3)$, i.e., a desired change in the pose e_t of the robotic end-effector. An individual low-level sub-task is identified by a *tool* denoted by o_t^+ and a *target* denoted by o_t^*, along with a way-point w_t. The tool is the object being grasped by the robot at time t, the target is the object to manipulate using the grasped tool and the predicted way-point is the desired pose of the tool in the target's frame at the end of the sub-task. The way-point w_t is a function of time as it changes based on the current pose of the tool relative to the target. Several way-points are often necessary to perform even simple tasks. For instance, in painting, a brush is the tool o_t^+ and a paint bucket is the target o_t^*. To load a brush with paint, several way-points in the bucket's frame need to be predicted. The first way-point can be when the brush touches the paint, while the second way-point is slightly above the paint. The tool o_t^+ and target assignment o_t^* are also functions of time t, and change as the system switches from one sub-task to the next, based on the current observation z_t and on what has been accomplished so far. For instance, after loading the brush, the robot switches to the next sub-task wherein the brush is still the tool object, but the painting canvas or surface becomes the new target object.

In the proposed model, observations are limited to $6D$ poses of objects and their semantic labels. These observations are often insufficient by themselves for determining the current stage of the task, for deciding to terminate the current sub-task and for selecting the next sub-task. For instance, in the painting example, the vision module does not provide information regarding the current status of the brush. Therefore, the robot needs to *remember* whether it has already dipped the brush in the paint. Since it is not practical to keep the entire sequence of past actions in memory, the approach uses a *Long Short-Term Memory* (LSTM) to compress the history h_t of the actions that the robot has performed so far, and use it as an input to the system along with observation z_t. The LSTM is trained along with the

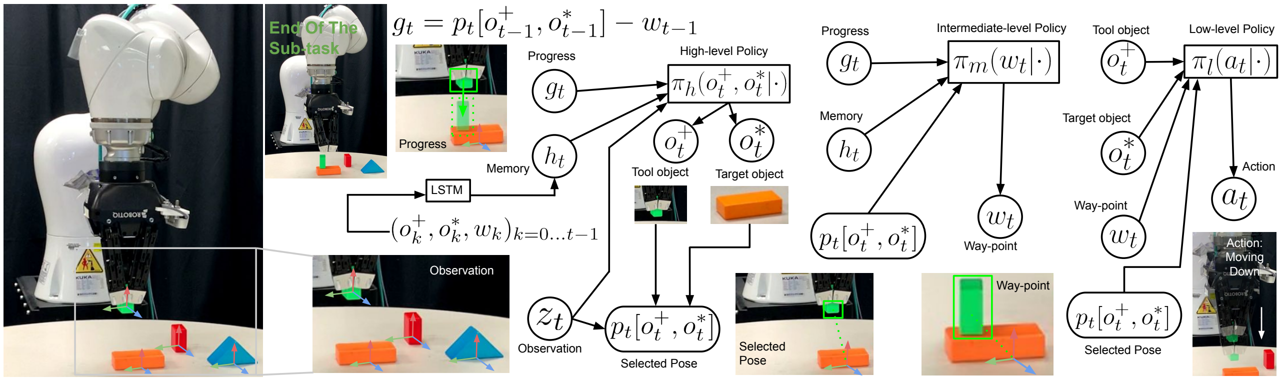

Fig. 2: Left: robot performing a stacking task. Right: policy networks. A high-level policy network uses the latest observation, current progress towards the current way-point and memory to select a pair of tool/target objects and their relative poses, which are then passed along to the intermediate policy for generating a next way-point. The low-level policy is responsible of generating the actual motion of the end-effector toward the way-point.

other parameters of the neural network.

The following describes the three levels of the hierarchical network architecture as depicted in Figure 2.

A **high-level** policy, denoted by π_h, returns a probability distribution over pairs (o^+_t, o^*_t) of objects, wherein o^+_t is the predicted tool at time t, and o^*_t is the predicted target at time t. The high-level policy takes as inputs 6D poses of objects and their semantic labels, along with the pose of the robot's end-effector. Additionally, the high-level policy π_h receives as inputs a history $h_t = (o^+_k, o^*_k, w_k)_{k=0,\dots,t-1}$ of pairs of tools and targets (o^+_k, o^*_k) and way-points w_k at different times k in the past, compressed into an LSTM unit, as well as a progress vector $g_t = p_t[o^+_{t-1}, o^*_{t-1}] - w_{t-1}$ that indicates how far is the tool o^+_{t-1} from the previous desired way-point w_{t-1} with respect to the target o^*_{t-1}.

An **intermediate-level** policy, denoted by π_m, receives as inputs the current tool o^+_t and target o^*_t, the pose $p_t[o^+_t, o^*_t]$ of o^+_t relative to o^*_t, in addition to history h_t and progress vector g_t. Both tool o^+_t and target o^*_t are predicted by the high-level policy π_h, as explained above. The intermediate-level policy returns a way-point $w_t \in \mathbb{R}^3 \times \mathbb{SO}(3)$, expressed in the coordinates system of the target object o^*_t.

A **low-level** policy, denoted by π_l, receives as inputs the current pose $p_t[o^+_t, o^*_t]$ of the current tool o^+_t relative to the current target o^*_t, in addition to the way-point w_t predicted by the intermediate policy, and returns a Gaussian distribution on action $a_t \in \mathbb{R}^3 \times \mathbb{SO}(3)$ that corresponds to a desired change in the pose e_t of the robotic end-effector.

IV. Learning Approach

In the proposed framework, an RGB-D camera is used to record a human performing an object manipulation task multiple times with varying initial placements of the objects. The pose estimation and tracking technique, explained in Section V-B, is then used to extract several trajectories of the form $\tau = (z_1, z_2, \dots, z_H)$, wherein $z_t = (e_t, \langle l^1_t, \dots, l^n_t \rangle, p_t)$ is the observed $6D$ poses of all objects at time t, including the end-effector's pose e_t. The goal of the learning process is to learn parameters of the three policy neural networks π_h, π_m and π_l that maximize the likelihood of the data τ and the inferred way-points, tools and targets, so that the system can generalize to novel placements of the objects that did not occur in the demonstrations. The likelihood is given by:

$$P\big((z_t, w_t, o^+_t, o^*_t)_{t=0:H}\big) = \Pi_{t=1}^{H-1} P_{A,t} P_{B,t}, \text{ with}$$

$$\begin{cases} P_{A,t} & \triangleq P(w_t, o^+_t, o^*_t | z_t) \\ & = \pi_h(o^+_t, o^*_t | z_t, h_t, g_t) \pi_m(w_t | h_t, g_t, p_t[o^+_t, o^*_t]), \\ P_{B,t} & \triangleq P(z_{t+1} | z_t, w_t, o^+_t, o^*_t) \\ & = \pi_l(e_{t+1} - e_t | w_t, o^+_t, o^*_t, p_t[o^+_t, o^*_t]), \end{cases}$$

wherein $h_t = (w_t, o^+_t, o^*_t)_{i=0}^{t-1}$ is the history and $g_t = p_t[o^+_{t-1}, o^*_{t-1}] - w_{t-1}$ is the progress vector.

The principal challenge here lies in the fact that the sequence $(w_t, o^+_t, o^*_t)_{t=0}^{H}$ of way-points, tools, and targets is unknown, since the proposed approach uses as inputs only 6D poses of objects at different time-steps and does not require any sort of manual annotation of the data.

To address this problem, an iterative learning process performed in three steps is proposed. First, the low-level policy is initialized by training on basic *reaching* tasks. The intermediate and high-level policies are initialized with prior distributions that simply encourage time continuity and proximity of way-points to target objects. Then, an expectation-maximization (EM) algorithm is devised to infer the most likely sequence $(w_t, o^+_t, o^*_t)_{t=0}^{H}$ of way-points, tools and targets in the demonstration data $(z_t)_{t=0}^{H}$. Finally, the three policy networks are trained by maximizing the likelihood of the demonstration data $(z_t)_{t=0}^{H}$ and the pseudo ground-truth data $(w_t, o^+_t, o^*_t)_{t=0}^{H}$ obtained from the EM algorithm. This process is repeated until the inferred pseudo ground-truth data $(w_t, o^+_t, o^*_t)_{t=0}^{H}$ become constant across iterations.

A. Prior Initialization

This section first explains how the low-level policy π_l is initialized. The most basic low-level skill is moving the end-effector between two points in $\mathbb{R}^3 \times \mathbb{SO}(3)$ that are relatively close to each other. We therefore initialize the low-level policy by training the policy network, using gradient-ascent, to maximize the likelihood of straight-line movements between consecutive poses e_{t+1} and e_t of the end-effector while aiming at way-points $\hat{w}_t \triangleq p_{t+1}[o^+_t, o^*_t]$. Therefore, the objective of the initialization process is given as $\max_{\theta_l} \sum_{t=1}^{H} \pi_l(e_{t+1} - e_t | o^+_t, o^*_t, p_t[o^+_t, o^*_t], \hat{w}_t)$, wherein each $\hat{w}_t$ is expressed in the frame of the target o_t^*, and θ_l are the parameters of the neural network π_l. Both o_t^+ and o_t^* are also chosen randomly in this initialization phase. The goal is to learn simple reaching skills, which will be refined and adapted in the learning steps to produce more complex motions, such as rotations.

The intermediate policy π_m is responsible for selecting way-point w_t given history h_t. It is initialized by constructing a discrete probability distribution over points $(\hat{w}_t, \hat{w}_{t+1}, \hat{w}_{t+2}, \dots, \hat{w}_H)$, defined as $\hat{w}_t \triangleq p_{t+1}[o_t^+, o_t^*]$. Poses p_{t+1} used as way-points $\hat{w}_t$ are obtained directly from demonstration data $(z_t)_{t=0}^H$. Specifically, we set $\pi_m(\hat{w}_k | h_t, o_t^+, o_t^*, p_t[o_t^+, o_t^*]) = 0$ for $k < t$, $\pi_m(\hat{w}_k | h_t, o_t^+, o_t^*, p_t[o_t^+, o_t^*]) \propto \exp(-\alpha \|\hat{w}_k\|_2)$ for $k = t$, and $\pi_m(\hat{w}_k | h_t, o_t^+, o_t^*, p_t[o_t^+, o_t^*]) \propto \exp(-\alpha \|\hat{w}_k\|_2) \frac{1-\beta}{H-t}$ for $k > t$, where α and β are predefined fixed hyper-parameters, and $\hat{w}_k$ is expressed in the coordinates system of the target o_t^*. This distribution encourages way-points to be close to the target at time t. This distribution is constructed for each candidate target $o_t^* \in \mathcal{O}$ at each time-step t, except for the robot's end-effector, which cannot be a target.

High-level policy π_h is responsible for selecting tools and targets (o_t^+, o_t^*) as a function of context. It is initialized by setting the tool as the object with the most motion relative to others: $o_t^+ = \arg\max_{o^i \in \mathcal{O} \setminus \{o^e\}} \sum_{o^j \in \mathcal{O}} \|p_{t+1}[o^i, o^j] - p_t[o^i, o^j]\|$, excluding the end-effector (or human hand) o^e. If all the objects besides the end-effector are stationary relative to each other, then no object is being used, and the end-effector is selected as the tool. Once the tool o_t^+ is fixed, the prior distribution on the target o_t^+ is set as: $\pi_h(o_t^+, o_t^* | h_t, p_t, g_t) = 0$ if $o_t^+ = o^e$ (the end-effector cannot be a target), $\pi_h(o_t^+, o_t^* | h_t, p_t, g_t) = \gamma$ if $o_t^+ = o_{t-1}^+$, and $\pi_h(o_t^+, o_t^* | h_t, p_t, g_t) = \frac{1-\gamma}{n-2}$ if $o_t^+ \neq o_{t-1}^+$, where o_{t-1}^+ is obtained from history h_t, n is the number of objects and γ is a fixed hyper-parameter, set to a value close to 1 to ensure that switching between targets does not occur frequently in a given trajectory.

B. Pseudo Ground-Truth Inference

After initializing π_h, π_m and π_l as in Section IV-A, the next step consists of inferring from the demonstrations $(z_t)_{t=1}^H$ a sequence $(o_t^+, o_t^*, w_t)_{t=1}^H$ of tools, targets and way-points that has the highest joint probability $P\big((z_t, w_t, o_t^+, o_t^*)_{t=0:H}\big)$ (Algorithm 1, lines 2-15). This problem is solved by using the *Viterbi* technique. In a forward pass (lines 2-12), the method computes the probability of the most likely sequence up to time $t-1$ that results in a choice (o_t^+, o_t^*, w_t) at time t. The log of this probability, denoted by $F_t[(o_t^+, o_t^*, w_t)]$, is computed by taking the product of three probabilities: (i) π_h: the probability of switching from o_{t-1}^+ and o_{t-1}^* as tools and targets to o_t^+ and o_t^*, given the progress vector g_t and the object poses relative to each other provided by the matrix p_t (which is obtained from observation z_t); (ii) π_m: the probability of selecting as a way-point a future pose $p_k[o_t^+, o_t^*]$ (denoted as w_k, $k \geq t$) for the tool relative to the target in the demonstration trajectory; this probability is also conditioned on choices made at the previous time step $t-1$; (iii) the likelihood of the observed movement of the objects at time t in the demonstration, given the choice (o_t^+, o_t^*, w_k) and the relative poses of the objects with respect to each other (given by matrix p_t). For each candidate (o_t^+, o_t^*, w_k) at time t, we keep in R_t the trace of the candidate at time $t-1$ that maximizes their joint probability. The backward pass (lines 13-15) finds the most likely sequence $(o_t^+, o_t^*, w_t)_{t=1}^H$ by starting from the end of the demonstration and following the trace of that sequence in R_t. The last step is to train π_m, π_l and π_h using the most likely sequence $(o_t^+, o_t^*, w_t)_{t=1}^H$ as a pseudo ground-truth for the tools, targets and way-points.

Algorithm 1: Learning Policies from Visual Demonstrations

```
Input: A set of $n$ objects $\mathcal{O} = \{o^1, o^2, \dots, o^n\}$; one or
       several demonstration trajectories $\{z_t\}_{t=1}^H$, wherein
       $z_t = (e_t, \langle l_t^1, \dots, l_t^n \rangle, p_t)$, $e_t$ is the end-effector's pose
       at time $t$, $p_t[o^i, o^j]$ is the 6D pose of $o^i \in \mathcal{O}$ relative to
       $o^j \in \mathcal{O}$, $\forall (o^i, o^j) \in \mathcal{O} \times \mathcal{O}$;
Output: High-level, intermediate-level, and low-level policies
        $\pi_h$, $\pi_m$ and $\pi_l$;
1  Initialize $\pi_h$, $\pi_m$ and $\pi_l$ (Section IV-A); $F_0[:] \leftarrow -\infty$;
2  for $t = 1; t \leq H; t \leftarrow t + 1$ do
3      foreach $(o_t^+, o_t^*, k) \in \mathcal{O} \times \mathcal{O} \times \{t, t+1, \dots, H\}$ do
4          $x_t \triangleq (o_t^+, o_t^*, k)$; $w_k \triangleq p_k[o_t^+, o_t^*]$;
5          $\Delta p_t \triangleq p_{t+1}[o_t^+, o_t^*] - p_t[o_t^+, o_t^*]$;
6          foreach $(o_{t-1}^+, o_{t-1}^*, k') \in \mathcal{O} \times \mathcal{O} \times \{t-1, \dots, H\}$ do
7              $x_{t-1} \triangleq (o_{t-1}^+, o_{t-1}^*, k')$; $w_{k'} \triangleq p_{k'}[o_{t-1}^+, o_{t-1}^*]$;
8              $h_t \leftarrow (o_{t-1}^+, o_{t-1}^*, w_{k'})$;
9              $g_t \leftarrow p_t[o_t^+, o_t^*] - w_t$;
10             $Q[x_{t-1}, x_t] \leftarrow \log\big(\pi_h(o_t^+, o_t^* | h_t, p_t, g_t)\big) +$
                 $\log\big(\pi_m(w_k | h_t, o_t^+, o_t^*, p_t[o_t^+, o_t^*])\big) +$
                 $\log\big(\pi_l(\Delta p_t[o_t^+, o_t^*] | o_t^+, o_t^*, p_t[o_t^+, o_t^*], w_k)\big) +$
                 $F_{t-1}[x_{t-1}]$;
11         $F_t[x_t] \leftarrow \max_{x_{t-1}} Q[x_{t-1}, x_t]$;
12         $R_t[x_t] \leftarrow \arg\max_{x_{t-1}} Q[x_{t-1}, x_t]$;
   /* Construct the most likely sequence */
13 $(o_H^+, o_H^*, k) \leftarrow \mathrm{argmax}_x\, R_t[x]$; $w_H \leftarrow p_k[o_H^+, o_H^*]$;
14 for $t = H - 1; t > 0; t \leftarrow t - 1$ do
15     $(o_t^+, o_t^*, k) \leftarrow \mathrm{argmax}_x\, R_{t+1}[x]$; $w_t \leftarrow p_k[o_t^+, o_t^*]$;
16 Train the policy networks $\pi_h$, $\pi_m$ and $\pi_l$ with $(o_t^+, o_t^*, w_t)_{t=1}^H$
     and $\{z_t\}_{t=1}^H$;
17 Optional: Go to 2 and repeat with updated policies $\pi_h$, $\pi_m$, $\pi_l$;
```

C. Training the Policy Networks

To train π_m, π_l and π_h using the pseudo ground-truth $(o_t^+, o_t^*, w_t)_{t=1}^H$, obtained as explained in the previous section, we apply the stochastic gradient-descent technique to simultaneously optimize the parameters of the networks by minimizing a loss function L defined as follows. L is defined as the sum of multiple terms. The first two are $L_{o^+} = \sum_t CE(\hat{o^+}_t, o_t^+)$ and $L_{o^*} = \sum_t CE(\hat{o^*}_t, o_t^*)$ where CE is the cross entropy, and $(\hat{o^+}_t, \hat{o^*}_t)$ is the current prediction of π_h. The third term is $L_w = \sum_t MSE(\hat{w}_t, w_t)$ where $\hat{w}_t$ is the output of π_m and MSE is the mean square error. The next term is $L_{action} = -log[\pi_l(a_t | p_t[o_t^+, o_t^*], w_t, o_t^+, o_t^*)]$, which corresponds to the log-likelihood of the low-level actions in the demonstrations. To further facilitate the training, two auxiliary losses are introduced. The first one is to encourage consistency within each sub-task. As the role of the memory in the architecture is to indicate the sequence sub-tasks that have been already performed, it should not change before (o_t^+, o_t^*, w_t) changes. If we denote the LSTM's output as $M(h_t)$ at time-step t, the consistency loss is defined as $L_{mem} = \sum_t \|M(h_t) - M(h_{t-1})\|_2 \times I_{o_{t-1}^+=o_{t-1}^+, o_{t-1}^*=o_t^*, w_{t-1}=w_t}$ where I is the indicator function. The last loss term, L_{ret}, is used to ensure that memory $M(h_t)$ retains sufficient information from previous steps. Thus, we train an additional layer $(ret_{o^*}, ret_{o^+}, ret_w)$ directly after $M(h_t)$ to retrieve at step t the target, tool and way-point of step $t-1$. L_{ret} is defined as $L_{ret} = CE(ret_{o^*}(M(h_t)), o_{t-1}^*) + CE(ret_{o^+}(M(h_t)), o_{t-1}^+) + MSE(ret_w(M(h_t)), w_{t-1})$. As a result, the complete proposed architecture is trained with the loss $L = L_{o^+} + L_{o^*} + L_w + L_{action} + L_{mem} + L_{ret}$.

V. Experimental Results

A. Data collection

We used an *Intel RealSense 415* camera to record several demonstrations of a human subject performing three tasks. The first task consists in inserting a paint brush into a bucket, then moving it to a painting surface and painting a virtual straight line on the surface. The poses of the brush, bucket and painting surface are all tracked in real-time using the technique explained in V-B. The second task consists in picking up various blocks and stacking them on top of each other to form a predefined desired pattern. The third task is similar to the second one, with the only difference being the desired stacking pattern. Additionally, we use the PyBullet physics engine to simulate a *Kuka* robotic arm and collect data regarding a fourth task. The fourth task consists in moving a wrench that is attached to the end-effector to four precise locations on a wheel, sequentially, rotating the wrench at each location to remove the lug-nuts, then moving the wrench to the wheel's center before finally pulling it.

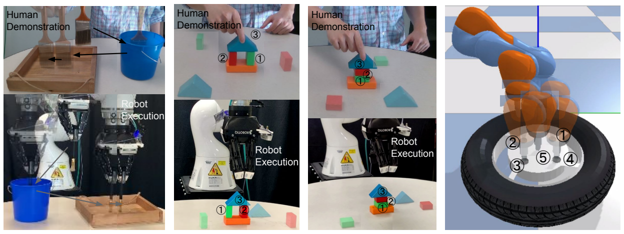

Fig. 3: Tasks considered in the experiments: painting (left), stacking (middle), and tire removal (right).

B. Object Pose Parsing from Demonstration Video

In each demonstration video, 6D poses $\xi \in SE(3)$ and semantic labels of all relevant objects are estimated in real-time and used to create observations $(z_t)_{t=1}^H$ as explained in Section III. Concretely, a scene-level multi-object pose estimator [28] is leveraged to compute globally the relevant objects' 6D poses in the first frame. It starts with a pose sampling process and performs Integer Linear Programming to

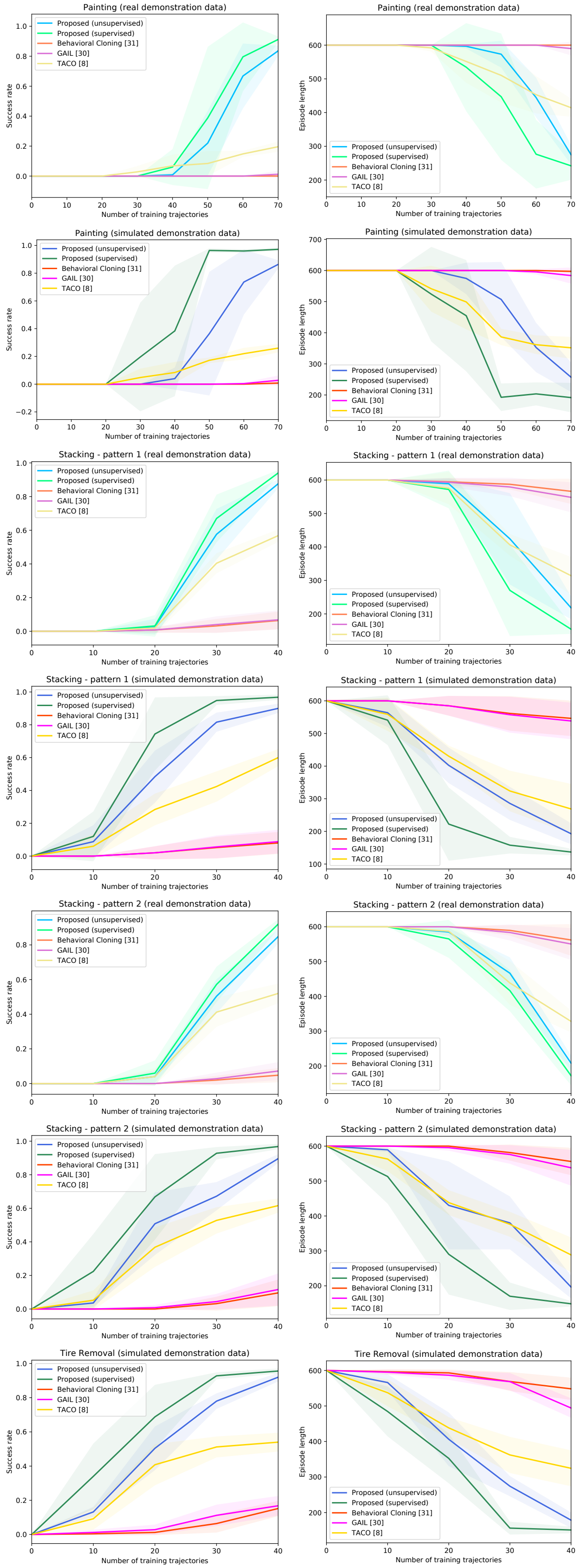

Fig. 4: Average success rate (left) and episode length (right) as a function of the number of training trajectories

ensure physical consistency by checking collisions between any two objects as well as collisions between any object and the table. Next, the poses computed from the first frame are used to initialize the se(3)-TrackNet [17], which returns a 6D pose for each object in every frame of the video. The 6D tracker requires access to the objects' CAD models. For the painting task, the brush and the bucket are 3D-scanned as in [29], while for all the other objects, models are obtained from CAD designs derived from geometric primitives. During inference on the demonstration videos, the tracker operates in $90Hz$, resulting in an average processing time of $13.3s$ for a 1 min demonstration video. The entire video parsing process is fully automated, and did not require any human input beyond providing CAD models of the objects offline to the se(3)-TrackNet in order to learn to track them.

C. Training and architecture details

The high-level, intermediate-level, and low-level policies are all neural networks. In the high-level policy, the progress vector g_t is embedded by a fully connected layer with 64 units followed by a ReLU layer. An LSTM layer with 32 units is used to encode history h_t. Observation z_t is concatenated with the LSTM units and the embedded progress vector and fed as an input to two hidden layers with 64 units followed by a ReLU layer. From the last hidden layer, target and tool objects are predicted by a fully connected layer and a softmax. The intermediate-level policy network consists of two hidden layers with 64 units. The low-level policy concatenates into a vector four inputs: 6D pose of the target in the frame of the tool object, way-point, and the semantic labels of the target and tool objects. After two hidden layers of 64 units and ReLU, the low-level policy outputs a Gaussian action distribution. The number of training iterations for all tasks is $20,000$, the batch size is $2,048$ steps, the learning rate is $1e-4$, and the optimizer is *Adam*. The hyper-parameters used in Section IV-A are set as $\gamma = 0.95$, $\alpha = 100$, and $\beta = 0.95$.

The proposed method is compared to three other techniques. Generative Adversarial Imitation Learning (**GAIL**) [30], Learning Task Decomposition via Temporal Alignment for Control (**TACO**) [8], and **Behavioral Cloning** [31] where we train the policy network of [30] directly to maximize the likelihood of the demonstrations without learning rewards. We also compare to a supervised variant of our proposed technique where we manually provide ground-truth tool and target objects and way-points for each frame. The supervised variant provides an upper bound on the performance of our unsupervised algorithm. Note also that TACO [8] requires providing manual *sketches* of the sub-tasks, whereas our algorithm is fully unsupervised.

D. Evaluation

Except for the tire removal, all tasks are evaluated using real demonstrations and a real *Kuka LBR* robot equipped with a *Robotiq* hand. The policies learned from real demonstrations are also tested extensively in the PyBullet simulator before testing them on the real robot.

	Painting	Stacking 1	Stacking 2
Proposed (unsupervised)	$5/5$	$5/5$	$5/5$
Behavioral Cloning [31]	$0/5$	$0/5$	$0/5$
GAIL [30]	$0/5$	$0/5$	$0/5$
TACO [8]	$1/5$	$1/5$	$2/5$

TABLE I: Success rates on the real Kuka robot

A painting task is successfully accomplished if the brush is moved into a specific region of a radius of $3cm$ inside the paint bucket, the brush is then moved to a plane that is $10cm$ on top of the painting surface, and finally the brush draws a virtual straight line of $5cm$ at least on that plane. A tire removal task is successfully accomplished if the robot removes all bolts by rotating its end-effector on top of each bolt (with a toleance of $5mm$) with at least 30° counter-clockwise, and then moves to the center of the wheel. A stacking task is successful if the centers of all the objects in their final configuration are within $0.5cm$ of the corresponding desired locations.

Figure 4 shows the success rates of the compared methods for the four tasks, as well as the length of the generated trajectories while solving these tasks in simulation, as a function of the number of demonstration trajectories collected as explained in Section V-A. The results are averaged over 5 independent runs, each run contains 50 test episodes that start with random layouts of the objects. Table I shows the success rates of the compared methods on the real Kuka robot, using the same demonstration trajectories that were used to generate Figure 4 (70 trajectories for painting and 40 for each of the remaining tasks). These results show clearly that the proposed approach outperforms the compared alternatives in terms of success rates and solves the four tasks with a smaller number of actions. The performance of our unsupervised approach is also close to that of the supervised variant. The proposed approach outperforms TACO despite the fact that TACO requires a form of supervision in its training. We also note that both our approach and TACO outperform GAIL and the behavioral cloning techniques, which clearly indicates the data-efficiency of compositional and hierarchical methods. Videos and supplementary material can be found at `https://tinyurl.com/2zrp2rzm`.

VI. Conclusion

We presented a unified neural-network framework for training robots to perform complex manipulation tasks that are composed of several sub-tasks. The proposed framework employs the principal of attention by training a high-level policy network to select a pair of tool and target objects dynamically, depending on the context. The proposed method outperformed alternative techniques for imitation learning, without requiring any supervision beyond recorded demonstration videos. While the current video parsing module requires the objects' CAD models beforehand, it is possible in future work to leverage model-free 6D pose trackers [32] for learning from demonstration involving novel unknown objects. We will also explore other applications of the proposed framework, such as real-world assembly tasks.

References

[1] S. Calinon, *Robot Programming by Demonstration*, 1st ed. USA: CRC Press, Inc., 2009.

[2] T. Osa, J. Pajarinen, and G. Neumann, *An Algorithmic Perspective on Imitation Learning*. Hanover, MA, USA: Now Publishers Inc., 2018.

[3] O. Kroemer and G. S. Sukhatme, "Learning spatial preconditions of manipulation skills using random forests," in *Proceedings of the IEEE-RAS International Conference on Humanoid Robotics*, 2016. [Online]. Available: http://robotics.usc.edu/publications/954/

[4] A. S. Wang and O. Kroemer, "Learning robust manipulation strategies with multimodal state transition models and recovery heuristics," in *Proceedings of (ICRA) International Conference on Robotics and Automation*, May 2019, pp. 1309 – 1315.

[5] Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez, "Learning compositional models of robot skills for task and motion planning," *Int. J. Robotics Res.*, vol. 40, no. 6-7, 2021. [Online]. Available: https://doi.org/10.1177/02783649211004615

[6] Z. Su, O. Kroemer, G. E. Loeb, G. S. Sukhatme, and S. Schaal, "Learning to switch between sensorimotor primitives using multimodal haptic signals," in *Proceedings of International Conference on Simulation of Adaptive Behavior (SAB '16): From Animals to Animats 14*, August 2016, pp. 170 – 182.

[7] H. Le, N. Jiang, A. Agarwal, M. Dudík, Y. Yue, and H. Daumé, "Hierarchical imitation and reinforcement learning," in *International conference on machine learning*. PMLR, 2018, pp. 2917–2926.

[8] K. Shiarlis, M. Wulfmeier, S. Salter, S. Whiteson, and I. Posner, "Taco: Learning task decomposition via temporal alignment for control," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4654–4663.

[9] G. Konidaris and A. Barto, "Skill discovery in continuous reinforcement learning domains using skill chaining," in *Advances in Neural Information Processing Systems*, Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, Eds., vol. 22. Curran Associates, Inc., 2009.

[10] M. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, S. Kraus, Ed. ijcai.org, 2019, pp. 6231–6235. [Online]. Available: https://doi.org/10.24963/ijcai.2019/869

[11] L. P. Kaelbling, "Learning to achieve goals," in *IN PROC. OF IJCAI-93*. Morgan Kaufmann, 1993, pp. 1094–1098.

[12] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *Proceedings of the 1st AAAI Conference on Bridging the Gap Between Task and Motion Planning*, ser. AAAIWS'10-01. AAAI Press, 2010, pp. 33–42.

[13] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," 2020.

[14] R. T. Icarte, T. Klassen, R. Valenzano, and S. McIlraith, "Using reward machines for high-level task specification and decomposition in reinforcement learning," ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. StockholmsmÃ€ssan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 2107–2116.

[15] R. Toro Icarte, E. Waldie, T. Klassen, R. Valenzano, M. Castro, and S. McIlraith, "Learning reward machines for partially observable reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 32, pp. 15 523–15 534, 2019.

[16] A. Camacho, R. T. Icarte, T. Q. Klassen, R. A. Valenzano, and S. A. McIlraith, "Ltl and beyond: Formal languages for reward function specification in reinforcement learning."

[17] B. Wen, C. Mitash, B. Ren, and K. E. Bekris, "se (3)-tracknet: Data-driven 6d pose tracking by calibrating image residuals in synthetic domains," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 10 367–10 373.

[18] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Scalable deep reinforcement learning for vision-based robotic manipulation," ser. Proceedings of Machine Learning Research, A. Billard, A. Dragan, J. Peters, and J. Morimoto, Eds., vol. 87. PMLR, 29–31 Oct 2018, pp. 651–673.

[19] R. Fox, R. Shin, S. Krishnan, K. Goldberg, D. Song, and I. Stoica, "Parametrized hierarchical procedures for neural programming," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rJl63fZRb

[20] D. Xu, S. Nair, Y. Zhu, J. Gao, A. Garg, L. Fei-Fei, and S. Savarese, "Neural task programming: Learning to generalize across hierarchical tasks." *CoRR*, vol. abs/1710.01813, 2017. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1710.html#abs-1710-01813

[21] D. Huang, S. Nair, D. Xu, Y. Zhu, A. Garg, L. Fei-Fei, S. Savarese, and J. C. Niebles, "Neural task graphs: Generalizing to unseen tasks from a single video demonstration," *CoRR*, vol. abs/1807.03480, 2018. [Online]. Available: http://arxiv.org/abs/1807.03480

[22] S. Nair, M. Babaeizadeh, C. Finn, S. Levine, and V. Kumar, "TRASS: time reversal as self-supervision," in *2020 IEEE International Conference on Robotics and Automation, ICRA 2020, Paris, France, May 31 - August 31, 2020*. IEEE, 2020, pp. 115–121. [Online]. Available: https://doi.org/10.1109/ICRA40945.2020.9196862

[23] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Curran Associates Inc., 2017, pp. 5055–5065.

[24] S. Nair and C. Finn, "Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation," *CoRR*, vol. abs/1909.05829, 2019. [Online]. Available: http://arxiv.org/abs/1909.05829

[25] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," ser. AAAI'17. AAAI Press, 2017, pp. 1726–1734.

[26] O. Nachum, S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18, 2018, pp. 3307–3317.

[27] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine, "Diversity is all you need: Learning skills without a reward function," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=SJx63jRqFm

[28] C. Mitash, B. Wen, K. Bekris, and A. Boularias, "Scene-level pose estimation for multiple instances of densely packed objects," in *Conference on Robot Learning*. PMLR, 2020, pp. 1133–1145.

[29] A. S. Morgan, B. Wen, J. Liang, A. Boularias, A. M. Dollar, and K. Bekris, "Vision-driven compliant manipulation for reliable, high-precision assembly tasks," *RSS*, 2021.

[30] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, pp. 4565–4573, 2016.

[31] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," Carnegie-Mellon University Pittsburgh PA, Tech. Rep., 1989.

[32] B. Wen and K. Bekris, "Bundletrack: 6d pose tracking for novel objects without instance or category-level 3d models," *IROS*, 2021.