Robotics and Autonomous Systems 61 (2013) 1323–1334

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

A syntactic approach to robot imitation learning using probabilistic activity grammars

Kyuhwa Lee *, Yanyu Su, Tae-Kyun Kim, Yiannis Demiris

Personal Robotics Lab, Department of Electrical and Electronic Engineering, Imperial College London, London SW7 2BT, UK

HIGHLIGHTS

- We present a syntactic approach to robot imitation learning.
- It captures reusable task structures in the form of probabilistic activity grammars.
- We aim to learn with a reasonably small number of samples under noisy conditions.
- We evaluate on both synthetic and two real-world humanoid robot experiments.
- Our method shows improvement on imitation learning when compared with other methods.

ARTICLE INFO

Article history:
Received 12 March 2013
Received in revised form
27 July 2013
Accepted 5 August 2013
Available online 19 August 2013

Keywords:
Robot imitation learning
Probabilistic grammars
Activity representation

ABSTRACT

This paper describes a syntactic approach to imitation learning that captures important task structures in the form of probabilistic activity grammars from a reasonably small number of samples under noisy conditions. We show that these learned grammars can be recursively applied to help recognize unforeseen, more complicated tasks that share underlying structures. The grammars enforce an observation to be consistent with the previously observed behaviors which can correct unexpected, out-of-context actions due to errors of the observer and/or demonstrator. To achieve this goal, our method (1) actively searches for frequently occurring action symbols that are subsets of input samples to uncover the hierarchical structure of the demonstration, and (2) considers the uncertainties of input symbols due to imperfect low-level detectors.

We evaluate the proposed method using both synthetic data and two sets of real-world humanoid robot experiments. In our Towers of Hanoi experiment, the robot learns the important constraints of the puzzle after observing demonstrators solving it. In our Dance Imitation experiment, the robot learns 3 types of dances from human demonstrations. The results suggest that under reasonable amount of noise, our method is capable of capturing the reusable task structures and generalizing them to cope with recursions.

1. Introduction

Humans are capable of learning novel activity representations despite noisy sensory input by making use of the previously acquired contextual knowledge, since many human activities often share similar underlying structures. For example, when we observe a hand transferring an object to another place where a grasping action cannot be seen due to some occlusions, we can still infer that a grasping action occurred before the object was lifted.

Similarly, in the process of language acquisition, a child learns more complex concepts and represents them by using previously learned vocabularies. Analogously, the structure of an activity can be represented using a formal grammar, where symbols (or vocabularies) represent the smallest meaningful units of action components, i.e. primitive actions. We are interested in learning reusable action components to better understand more complicated tasks that share the same structures under noisy environments.

The learning of reusable action components is one of the crucial tools for robot imitation learning (also called robot programming by demonstration), which has become an important paradigm, as it enables a robot to incrementally learn higher-level knowledge from human teachers. Our approach shares the concept of imitation learning presented in the Handbook of Robotics (Chapter 59) [1], as well as in [2–5] where a robot learns a new task directly from human demonstration without the need of extensive reprogramming.

There are several important issues in imitation learning: *what* to imitate, *how* to imitate, *who* to imitate, *when* to imitate and how to judge if imitation was successful [6]. In this paper, we mainly

* Corresponding author. Tel.: +44 7540328405.
E-mail address: lee.kyuh@gmail.com (K. Lee).

0921-8890/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.robot.2013.08.003

focus on the issue of *what* to imitate, which is an actively investigated area, where a robot needs to understand the goal or intention of actions, as done similarly in [7–11]. It is also known that humans tend to interpret actions based on goals rather than motion trajectories [12,13]. Another active research area, which studies on solving problems of *how* to imitate, focuses on learning the trajectories of joints (e.g. [14–19]). Although this is not our main focus, we address this issue in our Dance Imitation experiment (Section 5.3).

We are inspired by the work done in [20] which has the same motivation about hierarchical learning. In their work, the authors designed a set of primitive actions which are then used as building blocks, i.e. basic vocabularies, to represent higher-level activities. However, it does not deal with more complex concepts such as recursions which we will deal with here. In this respect, we choose Stochastic Context-Free Grammars (SCFGs) as our representation framework due to (1) robustness to noise as a result of the probabilistic nature, (2) compactness on representing hierarchical and recursive structures, and (3) generation of human-readable output which can be intuitively interpreted for users even without deep technical knowledge. It is worth noting that "context-free" in SCFG is used as a contrast to "context-sensitive", which is another type of grammars, i.e. it does not mean that it lacks the contextual knowledge. Although some other commonly used techniques such as Hidden Markov Models (HMMs) require lower computational complexity, they are often relatively less expressive, and cannot easily represent structures with repetitions and recursions. For example, the recursive activity $a^n b^n$, where a = Push, b = Pull (equal number of Push and Pull operations.), cannot be represented using HMMs. SCFGs extend Context-Free Grammars by adding rule probabilities, a notion similar to state transition probabilities in HMMs. We are especially interested in the real-world applications where noise cannot be avoided. Hence, in our case we consider the symbol probabilities as well as the rule probabilities.

In this paper, we present a method on learning activity grammars from human demonstrations which can be used as a prior to better recognize more complex tasks that share the same underlying components with ambiguity. We assume that (1) the system can detect meaningful atomic actions which are not necessarily noise-free, and (2) extensive complete datasets are not always available but numerous examples of smaller component elements could be found.

2. Related works

A large amount of effort has been spent to understand tasks using context-free grammars (CFGs). In [21], Ryoo defines a game activity representation using CFGs which enables a system to recognize events and actively provide proper feedback to the human user when the user makes unexpected actions. In [22], Ivanov defines SCFG rules to recognize more complicated actions, e.g. music conducting gestures, using HMM-based low-level action detectors. In [23], a robot imitates human demonstrations of organizing objects using SCFG-based task-independent action sequences. For other interesting areas that utilize CFGs as the underlying framework, e.g. computational biology and speech recognition, please refer to [24]. Aloimonos et al. [25] give the detailed explanations about various useful applications that use linguistic approaches including human motoric action representations.

The aforementioned studies consider cases where the proper grammar rules are given in advance. As opposed to manually defining the grammar rules to represent a task, there are also several approaches aiming at constructing (i.e. inducing) grammars from data. In an early work, Nevill-Manning et al. [26] presented the SEQUITUR algorithm which can discover the hierarchical structures among symbols. Solan et al. [27] presented the ADIOS algorithm which induces CFGs and context-sensitive

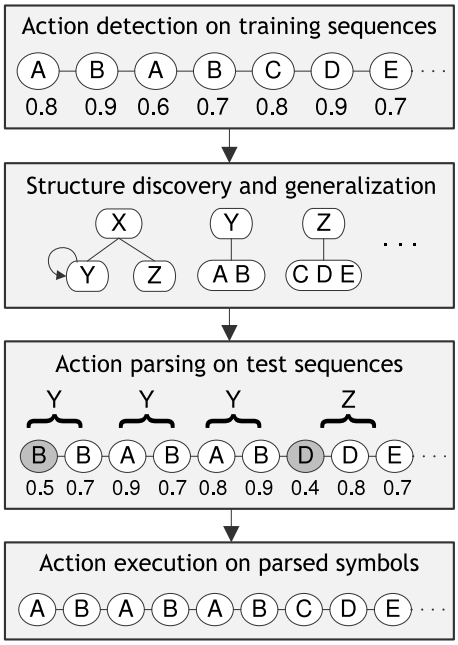


Fig. 1. Overview of our approach to imitation learning with an example. The input training sequences are converted into streams of symbols with probability, respectively indicated by circles and numbers below, from which the original structure is uncovered using grammatical representations. The acquired knowledge is used to better recognize unforeseen, more complex activities (test sequences) that share the same structure components.

grammars as well, with some restrictions (e.g. no recursions) using graphical representations. Stolcke and Omohundro [28] presented a SCFG induction technique, which more recently has been extended by Kitani et al. [29] to remove task-irrelevant noisy symbols to cope with more realistic environments. Lee et al. [30] apply SCFG learning algorithm to discover the optimal number of symbols required to represent a task. In [31], Ogale et al. construct a SCFG grammar based on frequency of human pose pairs, i.e. bigrams, considering slightly varying viewpoints. However, it does not have a generalization step which differs from our method.

Compared to the conventional learning techniques, our method has two distinctive features: (1) our method actively searches for frequently occurring substrings from the input stream that are likely to be meaningful to discover the hierarchical structures of activity; (2) we take into account the uncertainty values of the input symbols computed by low-level atomic action detectors. Fig. 1 gives an overview of our approach with an example for illustrative purpose. Similar to Ivanov's work [22] where they augmented the conventional SCFG "parser" by considering the uncertainty values of the input symbols, we extend the conventional SCFG "induction" technique by considering the uncertainty values of the input symbols.

In [28], Stolcke and Omohundro proposed a technique on merging states which generalizes SCFG rules to deal with unforeseen input with arbitrary lengths, e.g. symbols generated using recursive representations. They introduce two operators, chunking and merging, which convert an initial naive grammar to a more general one. The method assumes that input terminal symbols are deterministic, i.e. all symbols are equally reliable and do not contain any uncertainty values. Our method is different in that it takes into account the uncertainty (or probability) values of input symbols and explicitly searches for regularities using an n-gram-like frequency table within each input sample. This allows our method to learn a better grammar that reflects the noise term included in the observation.

More recently, Kitani et al. [29] presented a framework of discovering human activities from video sequences using a SCFG

induction technique based on [28]. By assuming that the noise symbols are not a part of the task representation, they try excluding some symbols from input stream until a grammar with strong regularity is found based on minimum description length (MDL) principle. However, since noise symbols are not assumed to be a part of task representation, this technique is limited to dealing with the insertion errors where wrong symbols are accidentally inserted.

In the human–robot interaction domain, Nicolescu and Mataric [32] presented a framework which generalizes the graph-based task representations by merging nodes to induce a graph with the longest common sequences. After learning, they allow their system to interactively modify the task representation from human vocal commands. The notion of nodes in their work corresponds to that of our non-terminal symbols which are essentially state representations. However, as their framework is inherently based on directional acyclic graphs, it cannot induce a representation containing recursive actions, which is often useful to describe periodic human movements.

3. Background

3.1. Stochastic context-free grammar induction

A context-free grammar (CFG) is defined by a 4-tuple $G = \{\Sigma, N, S, R\}$, where Σ is the set of terminals, N is the set of non-terminals, R is the set of production rules, and S is the start symbol. The production rules take the form $X \rightarrow \lambda$, where $X \in N$ and $\lambda \in (N \cup \Sigma)^*$. Non-terminals are denoted in uppercase letters while terminals are denoted in lowercase letters. In Stochastic CFG (SCFG), also known as Probabilistic CFG (PCFG), each rule production is assigned continuous probability parameters.

To induce an activity grammar from the input data (terminal symbols), first an initial naive grammar is built as the starting point by adding all input sequences to the start symbol S. Starting from the initial grammar, two kinds of operators, *Substitute* and *Merge*, are applied until the grammar is found. The quality of a grammar is measured by the Minimum Description Length (MDL) principle as used in [33,29,28], which will be explained more in Section 3.2. In the context of robot imitation learning of human tasks, the technique of merging repetitive symbols used in [34] can be reinterpreted as a means of abstracting meaningful actions into hierarchical structures.

There are two operators that abstract and generalize the initial grammar. The *Substitute* operator builds hierarchy by replacing a partial sequence of symbols in the right-hand side of a rule with a new non-terminal symbol. The new rule is created such that a new non-terminal symbol expands to these symbols. The *Merge* operator generalizes rules by replacing two symbols with the same symbol. *Merge*(X, Y) *into* Z means all X and Y symbols in production rules are replaced with the symbol Z. As a result, it converts the grammar into the one that can generate (or accept) more symbols than its predecessor while reducing the total length of the grammar.

The challenging problem here is that there is no obvious way to efficiently choose which operator to apply. In the case of HMMs, choosing the locally best choice (greedy strategy) generally leads to good results [28]. However, it is no longer the case in SCFGs as *Substitute* operator often requires several following *Merge* or *Substitute* operators to produce a better grammar. In his original work [28], Stolcke uses the beam-search method to limit the search space, which considers a number of relatively good grammars in parallel and stops if certain neighborhood of alternative models has been searched without producing further improvements. We use the beam search strategy with depth 3, which is reported to find most of the important grammatical structures of SCFG [28].

3.2. Measuring the quality of a grammar

Our goal is to find a grammar that is sufficiently simple yet expressive as pointed out by Langley et al. [33]. In his work, a minimum-description length (MDL) principle is used to decide whether or not to merge states.

We denote $P(M)$ as *a priori* model probability, where M is a grammar model that includes structure priors $P(M_S)$ and parameter priors $P(M_\theta)$ that do not consider the input data D, where $P(D|M)$ denotes a data likelihood.

$$P(M) = P(M_S, M_\theta) = P(M_S)P(M_\theta|M_S). \tag{1}$$

Where $P(M_S)$ specifies the structure prior, i.e. the length of a grammar, and $P(M_\theta)$ specifies the parameter prior, i.e. rule probabilities. Maximizing the joint probability $P(M, D)$

$$P(M, D) = P(M)P(D|M) \tag{2}$$

is equivalent to minimizing

$$-\log P(M, D) = -\log P(M) - \log P(D|M) \tag{3}$$

where $-\log P(M)$ represents the description length of the model under the given prior distribution and $-\log P(D|M)$ represents the description of the data D given a model M. The sum of two negative log values naturally corresponds to the total description length of the model and data. Thus, the goal can be rephrased as minimizing $-\log P(M, D)$.

Although one can define the prior distribution of $P(M_S)$ in a simple form such as e^{-l}, where l = number of bits required to encode the grammar, it is far from being a natural distribution for grammars. Thus, a Poisson distribution is commonly used with a mean of 3.0 (average production length) as in [28,29].

The data likelihood $P(D|M)$ is computed using Viterbi parsing, which is commonly used in HMMs. However, unlike [28,29], to handle the uncertainty values of the input symbols, the method of computing the likelihood needs to be modified. To cope with this situation, we use the SCFG parsing algorithm with uncertainty input introduced in [22] to compute data likelihood.

4. Proposed method

We first explain our method of computing the rule probabilities in the first section, followed by considering symbols with uncertainty values.

4.1. Active substring discovery

In our framework, each terminal symbol represents a primitive action unit which contains a probability value, i.e. the symbol detector confidence. Each non-terminal symbol represents an abstraction of terminal symbols.

To generate a grammar that focuses on patterns with strong regularity, we build an n-gram-like frequency table which keeps the number of occurrences of substrings that are subset of input sequences. The score of a rule $X \rightarrow \lambda$ is the occurrence value of λ in the frequency table multiplied by the expected probability value of λ. Its calculation will be discussed in Section 4.2. This is different from [29] where they use a similar table to choose the best candidate symbols which has the maximum compression rate for *Substitute* operation discussed in Section 3.1.

For simplicity, we first consider the case without uncertainty values. In this case, as defined in [28,29], the rule probability is calculated by normalizing rule scores, i.e.:

$$P(X \rightarrow \lambda_i) = \frac{f(X \rightarrow \lambda_i)}{\sum_k f(X \rightarrow \lambda_k)} \tag{4}$$

a
S→ABABABAB (6) [0.86]
| ABACABAB (1) [0.14]

b
S→ZZ (6) [0.86]
| XYZ (1) [0.14]
X→AB (27) [1.00]
Y→AC (1) [1.00]
Z→XX (13) [1.00]

c
S→ZZ (7) [1.00]
Z→AB (27) [0.66]
| AC (1) [0.02]
| ZZ (13) [0.32]

d
S→SS (20) [0.42]
| AB (27) [0.56]
| AC (1) [0.02]

Fig. 2. A simple example that shows how the task structure is discovered and generalized. (a) Initial naive grammar. (b) After substituting *AB* with *X*, *AC* with *Y*, and *XX* with *Z*. (c) After merging (X, Y) to *X* and merging (X, Z) to *Z*. (d) After merging (S, Z) to *S*. Please note that the uncertainties of symbols are not considered in this example for simplicity, where Eq. (4) is used instead of Eq. (8).

where λ_i is the *i*th rule production of non-terminal *X* and $f(\cdot)$ denotes the frequency of the string. $P(X \rightarrow \lambda_i)$ satisfies the following property:

$$\sum_i P(X \rightarrow \lambda_i) = 1. \tag{5}$$

In our method, as we keep counts for all possible sub-patterns from input samples, the probability of each rule is always larger than zero even if there was no input sequence that exactly matches the discovered sub-pattern. This has an effect of stronger "inductive leap", i.e. higher tendency to generalize from a relatively small number of input samples.

To illustrate, suppose that we want to learn an activity with repetitions $(ab)^n$ from the 6 correct samples of "*abababab*" and 1 erroneous sample of "*abacabab*". The initial naive grammar (Fig. 2(a)) simply contains all input sequences. We use parentheses $(\cdot)$ and brackets $[\cdot]$ to represent counts and probability values, respectively, e.g. $S \rightarrow ABC$ (20) [0.90] represents the rule score of 20 and rule probability 0.90. We now apply a *Substitute* (Fig. 2(b)) and *Merge* operators (Fig. 2(c)–(d)) introduced in [28] with rule scores obtained from our frequency table. Fig. 2(a) shows an initial naive grammar. After *Substituting AB* with *X*, *AC* with *Y*, and *XX* with *Z*, we obtain the grammar in Fig. 2(b). After *Merging* (X, Y) to *X*, *Merging* (X, Z) to *Z*, and finally *Merging* (S, Z) to *S*, we obtain the grammar in Fig. 2(d).

We have now obtained a more generalized grammar that favors (yielding higher probability when parsed) input sequences mostly containing *AB*'s. It is worth noting that the rule probability of erroneous symbol *AC* is still in the grammar but with very low probability. As a result, this grammar "allows" occasional errors as it still accepts noise cases with low probability instead of simply rejecting the whole observation sequence. This ability to make a soft decision is one of the advantages of SCFGs, when compared to non-stochastic CFGs which do not have rule probability values.

In practice, it is often useful to limit the maximum length of symbols to be considered in the frequency table to avoid generating an exhaustive list of symbols to increase the speed. This is a reasonable assumption as human activities usually involve repetitive action components [35]. Also, considering only the n-most frequent substring patterns is an effective alternative. Since the search space of the possible grammars is not small, a beam search strategy is applied as in [28] which considers a number of relatively good grammars in parallel and stops if a certain neighborhood of alternative grammar models has been searched without producing further improvements.

4.2. Considering input samples with uncertainty

So far, we have only considered a case where the input symbols are non-probabilistic, i.e. terminals $(a, b, c \ldots)$ are not assigned with probability values. However, since we assume that low-level action detectors could also provide uncertainty (confidence) values as output, it is beneficial to exploit this information. If there is a higher rate of noise, it is more likely that the certainty of a symbol is lower. Based on this assumption, we first compute the probability of a sub-pattern $\boldsymbol{\lambda} = s_1 s_2 s_3 \ldots s_l$ of length *l* from input, as

$$P(\lambda) = \left(\prod_n P(s_l) \right)^{\frac{1}{l}}. \tag{6}$$

The term $\frac{1}{l}$ is used to normalize the likelihood, since the probability is in overall multiplied by the number of symbols at the end of the parsing. The expected value of λ, $\mu(\lambda)$, is obtained by averaging all occurrences of $\lambda = \{\lambda_1, \lambda_2 \ldots\}$ in the input, i.e.:

$$\mu(\lambda) = \frac{1}{n} \sum_{k=1}^{n} P(\lambda_k). \tag{7}$$

Here, $\{\lambda_1, \lambda_2 \ldots\}$ are the same strings with possibly different probabilities, since each λ_i was sampled at different times by the detector. In prior works [28,29], $\mu(\lambda)$ is computed directly by the occurrences of λ, whereas in our case, we take into account the confidence values of detectors. Thus, we modify Eq. (4) as

$$P(X \rightarrow \lambda_i) = \frac{f(X \rightarrow \lambda_i)\mu(\lambda_i)}{\sum_k f(X \rightarrow \lambda_k)\mu(\lambda_k)} \tag{8}$$

where the subscript *i* denotes the *i*-th rule of *X*. We use this equation throughout our experiments. The above rule probability is used to compute the maximum likelihood in our Bayesian framework to update the MDL score (Eq. (2)).

In our method, we define the model prior probability

$$P(M) = P(M_S, M_\theta) = P(M_S)P(M_\theta|M_S) \tag{9}$$

where $P(M_S)$ denotes structure prior and $P(M_\theta|M_S)$ denotes the parameter prior, which depends on the structure. $P(M_S)$ is defined as Poisson distribution with mean (average production length) 3.0, as in [28,29]. The higher mean value means that the expected length of a production rule is larger.

In the literature, it is a common choice to model the parameter prior $P(M_\theta|M_S)$ as a symmetric Dirichlet distribution for grammar induction, e.g. [36,37,28,29]. This is because Dirichlet prior is a conjugate prior, which allows the posterior distribution to be a simple product of the prior and the likelihood. The parameter prior is the product of Dirichlet distributions, each of which corresponding to the prior distribution over possible *n* expansions of a single non-terminal *X*:

$$P_X(M_\theta|M_S) = \frac{1}{\beta(\alpha_1, \ldots, \alpha_n)} \prod_{i=1}^{n} \theta_i^{\alpha_i - 1}, \tag{10}$$

where β, the normalizing factor, is a multinomial Beta function with parameters α_i, and θ_i is a rule prior which is uniformly distributed. Since we have no prior knowledge about the distribution of the parameters, $\alpha_i = \alpha_j \forall i, j$ and $\sum_i^n \alpha_i = 1$.

Here, we briefly discuss about the effect of the values of α in a symmetric Dirichlet distribution, where $\alpha_i = \alpha_j \ \forall i, j$, while computing the MAP estimates. If $\alpha_i > 1$, the resulting grammar tends to have rule probabilities that are more equally likely as α_i gets larger, even if the rule probabilities computed in Eq. (8) are biased. If $\alpha_i < 1$, the rule probabilities tend to spread out towards extremes (0 or 1), where this tendency becomes stronger as α_i reaches towards zero ($\alpha_i > 0$). Lastly, if $\alpha_i = 1$, there is no prior on the distribution of rule probabilities, thus depending only on the rule probabilities computed by Eq. (8).

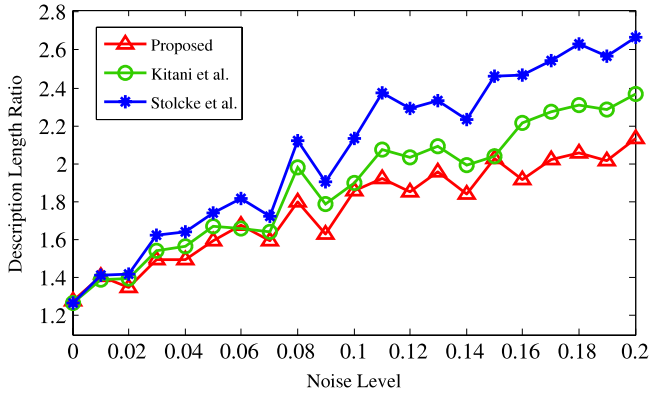


Fig. 3. Description length ratios of grammars generated by different methods. The lower score indicates that the grammar is more compact yet maintains sufficient expressive power.

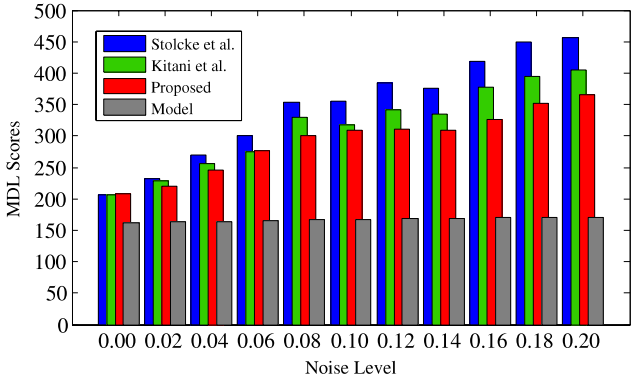


Fig. 4. Actual MDL scores for each method compared with the model grammar. MDL scores are averaged over 10 trials for each noise condition. The graph is shown with a 2% step for better view. A lower score indicates that the grammar is more compact yet reasonably expressive. How these scores affect the performance in the real world will be discussed in Sections 5.2 and 5.3.

We apply a pruning process as in [28] to speed up the induction and filter out non-critical production rules having probabilities lower than a certain threshold τ, as they are often accidentally created due to noise. If the removal of a rule decreases the description length of model prior but increases that of data likelihood in relatively small amount, it will lead to a better (lower) MDL score. We set $\tau = 0.01$ in all of our experiments. However, we later experimentally show the pruning effect in Section 5.2.3, by varying the threshold value.

5. Experiments and analyses

To test our framework, we first experiment on the synthetic data with systematically varying the levels of noise, followed by the real-world data obtained from a camera. As MDL scores depend on the data samples, we compute the ratio values of MDL scores between the learned grammar and the hand-made model grammar.

5.1. Bag-of-balls experiment

In this experiment, we assume a scenario where an arbitrary number of balls is put into a bag (denoted as *a*), moved to another place (denoted as *b*), and the same number of balls is taken out later (denoted as *c*), which can be represented in the form $a^n bc^n$. The samples are randomly generated from this model grammar up to the length of 9 ($n = 4$).

To test over noise sensitiveness, we add *Insertion* and *Substitution* errors. An *Insertion* error inserts a random symbol into the input and a *Substitution* error randomly replaces a symbol with any incorrect one. We test with the noise probability in the range of (0%, 20%) with 1% step, totaling in 21 noise conditions. A noise probability of 10% means that either a *Substitution* or *Insertion* error has occurred in approximately 10% of the input symbols. Each noise condition is conducted 10 times with randomly generated dataset and its mean MDL score is computed, resulting in 210 experiments in total. We compare the results using our method and two previously reviewed methods proposed by Kitani [29] and Stolcke [28].

The confidence values of terminal symbols are given such that the correct symbol is assigned with the probability computed from Gaussian distribution with $\mu = 0.85, \sigma = 0.1$ and wrong symbol with $\mu = 0.15, \sigma = 0.1$. We set unrelated symbol *d* to be included as noise, as in [29].

The description length ratio of a grammar is the ratio of MDL scores between learned grammar and the model grammar, where the lower score indicates that the grammar is more compact yet maintains enough expressive power. Fig. 3 shows the description length ratios over various noise conditions, where in most cases the grammars generated by our proposed method have the lowest description length ratio implying that they are well-balanced between compactness and expressiveness. We prune the production rules that are less than 1%, which are generally obtained due to noise.

As qualitative analysis, we now examine some of the obtained grammars. In the case with noise probability 0.08, a grammar obtained using the method proposed in [29] is shown in Fig. 5(a). Under this noise condition, the mean MDL score was 330.38 and the standard deviation was 39.72. A grammar obtained using our proposed method under the same noise condition with the same dataset is shown in Fig. 5(b). The mean MDL score was 300.62 and the standard deviation was 48.27. The average MDL scores can be seen in Fig. 4.

It is worth noting that the rule scores in the grammar generated using our method reflect the uncertainty values of input symbols. This is reflected in Fig. 5(b), where the erroneous sequence *AABAC*, which is supposed be *AABCC*, has a rule score of 0.46 in contrast to 1.00 in Fig. 5(a). This is because the fourth symbol, *A*, had a low probability (high uncertainty) due to noise. In the second grammar, since rules containing noise quickly converged to very low probability (less than 0.01) and pruned, the rule probability for the correct cases, e.g. $S \rightarrow AABCC$ has a relatively higher probability value. This will result in higher likelihood when parsed on new samples that contain the same pattern.

In the following section, we show how MDL scores actually reflect the performance in several real world robot scenarios.

5.2. The towers of Hanoi

We evaluate our method on the real-world data obtained from the demonstrations of 5 human participants using a camera. The aim of this experiment is to make a robot correctly imitate human action sequences that leads to successfully executing a task. However, instead of simply imitating, we require that the robot should deal with the noise using the previously obtained knowledge so that it can perform the intended action sequence correctly even when the perceived actions are partially incorrect. Furthermore, we are interested in challenging tasks that include recursion which can be demonstrated with various lengths of action sequences. We choose the Towers of Hanoi problem as it satisfies the above requirements. As discussed in Section 1, we tackle the problem from the "*what* to imitate" perspective, i.e. at the symbolic level rather than trajectory level. Thus, it is worth noting that we represent in this experiment each symbol as action goal, e.g. LIFT AN OBJECT, rather than trajectories.

5.2.1. Experiment scenario

In the training phase, a human demonstrator shows solving the puzzle using 2 and 3 disks, respectively, repeating each task

a

S→Y		(8.00)	[0.53]
	|AYC	(3.00)	[0.20]
	|AABAC	(1.00)	[0.07]
	|AACACCCC	(1.00)	[0.07]
	|AAYCC	(1.00)	[0.07]
	|CY	(1.00)	[0.07]
Y→AABCC		(8.00)	[1.00]

b

S→AABCC		(6.99)	[0.60]
	|ASC	(2.66)	[0.23]
	|AASCC	(0.93)	[0.08]
	|CS	(0.64)	[0.05]
	|AABAC	(0.46)	[0.04]

Fig. 5. (a) Obtained grammars using the method in [29] and (b) the proposed method from data with noise probability 0.08.

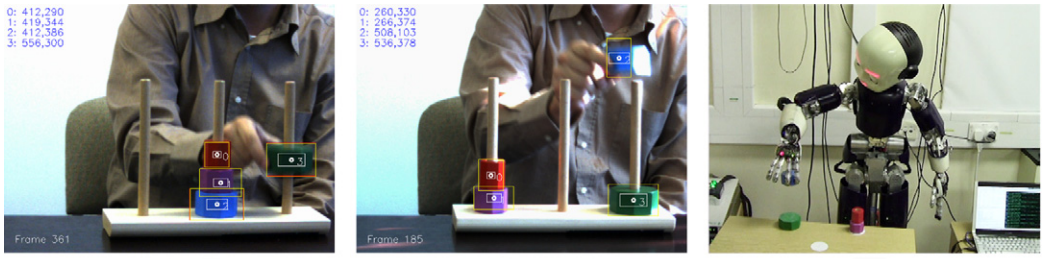


Fig. 6. (a)–(b) A sample tracking screen while a human participant is solving the puzzle with 4 disks. Compared to the low-noise condition (a), the high-noise condition (b) shows overexposed spots which often makes the tracker unstable. The tracker immediately resets the position if lost by searching the desired blob from the entire region of the image. (c) shows iCub performing parsed actions. A demo video is available at: http://www.youtube.com/watch?v=S99ViThK050.

3 times. The robot then learns an activity grammar from each demonstrator using techniques explained in Section 4. Thus, 5 activity grammars are learned in total.

In testing phase, a human demonstrator solves the puzzle using 4 disks, repeating 3 times. The trained activity grammar is used to parse the observation, which generates a sequence of actions to execute. A trial is considered a success only if the robot solves the puzzle by correctly executing the complete sequence of actions. Each activity grammar is used to parse each demonstration, which results in 15 tests for each of our 5 participants, or 75 in total. We use the iCub [38], a humanoid robot with 53 degree of freedom, as our testing platform. Fig. 6(c) shows a sample image of iCub executing the parsed actions.

We experiment under two types of noise conditions: the low-noise (indoor lighting) and high-noise (direct sunlight) conditions. That is, (a) train on the low-noise condition and test on both low- and high-noise conditions, respectively, and (b) train on the high-noise condition and test on both conditions. All samples of the high-noise dataset were captured in the same day for consistency. Example samples can be seen in Fig. 6.

Since we are interested in high-level task representations, we assume that the system can detect minimal level of meaningful actions and generate symbols. Similar to [22], we define these atomic action detectors using HMMs where each model corresponds to an action symbol with its output value representing the symbol's certainty, or probability value. The input to these detectors are the currently moving object's quantized direction, and distances between the object and towers.

In this experiment, our system generates 5 types of action symbols during an observation as detailed in Fig. 7. The reason we define symbols like *Disk moved "between" A and B* instead of *Disk moved "from" A to B* is because they are sufficient to represent the task structure without generating an excessive number of symbols. As the rule of the puzzle enforces that only a smaller disk shall be placed on top of the bigger disk, there is always only a single possibility of moving a disk between two towers. This is a fair assumption as this rule is always given in prior, not learned. Thus, in terms of executing symbols *A*, *B*, and *C*, we can expect that the robot will make the correct move. During the training phase, the symbol with the highest certainty is fed into the input of our proposed method (Section 4).

Symbol	Actions
L	Lift a disk
D	Drop a disk
A	Move between 1 and 2
B	Move between 1 and 3
C	Move between 2 and 3

Fig. 7. Actions defined in *Towers of Hanoi* experiment. The system is equipped with these 5 primitive action detectors which generates symbol probability during observation.

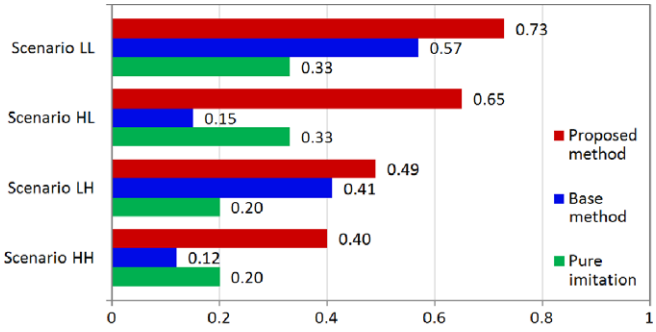


Fig. 8. Success rates using our method, base method [28] and the pure imitation. Scenarios LL and LH: train on the low-noise condition and test on low- and high-noise conditions, respectively. Scenarios HL and HH: train on high-noise condition and test on low- and high-noise conditions, respectively. The fact that a single mistake while parsing a long test sequence causes a failure makes this problem non-trivial.

If we denote action sequences *LAD* as *X*, *LBD* as *Y*, and *LCD* as *Z*, then symbols *X*, *Y*, and *Z* represent *pick-and-place* action sequences. The optimal solution of the puzzle can be represented as $((LAD)(LBD)(LCD))^n$, or $(XYZ)^n$, meaning "Perform (XYZ) recursively until the problem is solved".

We use a color camera with resolution 640 × 480, 30 frames per second. Object trackers are implemented using the standard CamShift algorithm provided in [39], with additional Kalman filtering to improve stability. A sample tracking screen is shown in Fig. 6; as it depends on the color information of blobs, it often produces errors due to lighting conditions. In addition to detection errors, it

Scenario	Method	Success	Avg.MDL	Scenario	Method	Success	Avg.MDL
LL	Proposed	55	284.63	LH	Proposed	37	286.92
	Base	43	390.28		Base	31	393.26
	Pure Imi.	25	N/A		Pure Imi.	15	N/A
HL	Proposed	49	306.25	HH	Proposed	30	306.66
	Base	11	469.32		Base	9	469.46
	Pure Imi.	25	N/A		Pure Imi.	15	N/A

Fig. 9. Detailed results with average MDL scores for comparison. Each case is tested on 75 sequences. MDL score is not available for the pure imitation as it does not rely on any learned model. It is worth noting that lower MDL scores generally lead to higher success rates.

Demonstrations using 4 disks	Low-noise	High-noise	Total
Total number of sequences	15	15	30
Sequences containing wrong symbol	10	12	22
Average number of error symbols per trial	1.13	2.20	1.67

Fig. 10. Error statistics of demonstrations using 4 disks on each noise condition. Note that even in the low-noise condition, there are only 5 trials observed with all correct symbols, which means that in most cases the pure imitation will not lead to the desired goal state. Each testing sequence is composed of 45 action symbols, which makes this problem non-trivial as only a single mistake will make it fail to achieve the goal.

is important to mention that our training and testing dataset also include human errors.

We use the standard Cartesian control library developed by Pattacini et al. [40] and a grasp trajectory planning method reported in [41] to execute the Tower of Hanoi task on iCub. We use this method to effectively deal with the position errors of disks, which internally uses a grasp simulator to plan the optimal trajectory of hand joints for every disk.

5.2.2. Results and discussions

As explained in the last section, the objective here is to learn a high-level task representation from a few short sequences of demonstrations that can be used to better parse unforeseen, possibly more complicated activities that share of same action components.

To evaluate the effectiveness of our method, we run the experiment in 4 different scenarios. In scenarios LL and LH, models are both trained from demonstrations of 2 and 3 disks under the low-noise condition, then they are tested on demonstrations of 4 disks on the low-noise (LL) and high-noise (LH) conditions, respectively. Similarly, scenarios HL and HH are both trained from the high-noise condition and tested on the low-noise (HL) and high-noise (HH) conditions, respectively. We report the results in Fig. 8.

We compare with the base method [28] and the pure imitation method which simply follows what has been observed from demonstrations. In any case, if the system makes any single mistake while recognizing human demonstration due to either wrong tracking or wrong symbol interpretation, it is marked as failed. This makes our scenarios non-trivial as each testing sequence is composed of 45 symbols. Please refer to Fig. 10 to see the error statistics. We do not use the method proposed by Kitani et al. [29] in this experiment as all generated symbols are always related to the task.

As can be seen in Fig. 8, it is important to note that there is a noticeable difference in the base method between scenarios LL and HL, and between LH and HH. As scenarios HL and HH are trained from noisy training data, the task representations could be easily corrupted. This could even lead to parse the correct symbol into wrong symbol which results in worse performance than purely imitating observed actions, whereas our method at least performs better than the pure imitation.

Fig. 11 shows a test example with 4 disks, where some of the ambiguous observations are clarified using the learned grammars at the parsing time. Fig. 11(a) shows where the block is being dropped (symbol D). Due to tracker error, the certainty of symbol A was higher than symbol D. It was disambiguated and corrected at the parsing time, as shown in Fig. 11(b).

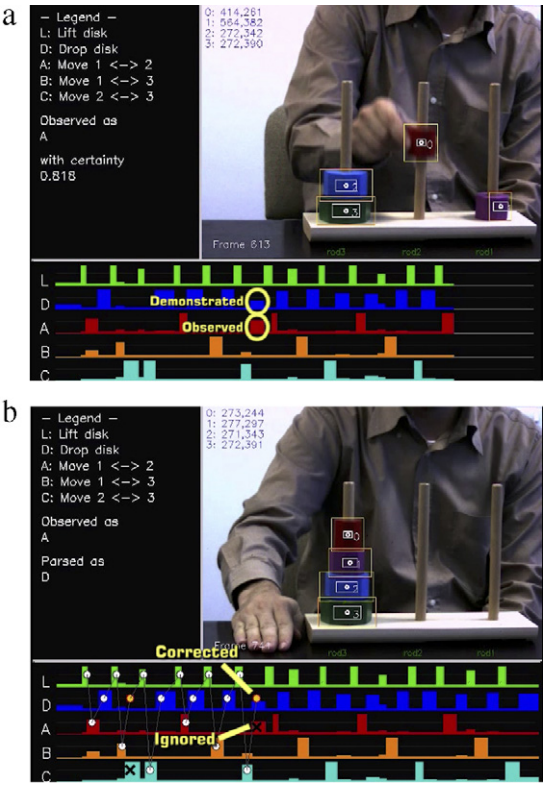


Fig. 11. Test example with 4 disks. (a) A participant demonstrates solving the puzzle using 4 disks, where the block is being dropped (symbol D). Due to tracker error, the certainty of symbol A was higher than D. (b) Ambiguous symbols are corrected at the parsing time.

It is also worth noting that from Fig. 9, we can confirm that lower MDL score leads to generally better representations. A model with the highest MDL score 469.46 (scenario HH, Base method) had the poorest performance, where a model with the lowest MDL score 284.63 (scenario LL, Proposed method) exhibited the best performance. As expected, models learned in the high-noise condition tend to have lengthier descriptions, which increases prior score. Relatively high MDL scores generally mean that they are too specific, failing to capture the recursiveness nature of the task.

The example grammar constructed using the proposed method (Fig. 12(a)) shows that it captured meaningful action components: *LAD*, *LBD*, and *LCD* (lines 1–3). Although there are intermittent error

a
S→LAD [0.205669]
| LBD [0.204606]
| LCD [0.163233]
| CADSS [0.020551]
| SLBAS [0.017184]
| SSS [0.388758]

b
S→LADLBDLCD [0.666667]
| SSLAD [0.285714]
| SSSSS [0.047619]

c
S →LADLBDX [0.601507]
| LXCX [0.248180]
| SSLAD [0.150313]
X→ADLL [0.200956]
| LCD [0.799044]

Fig. 12. (a) A sample grammar that captured the meaningful action components such as *LAD*, *LBD*, and *LCD* (lines 1–3). These components can be used to enforce the observation to be consistent with the demonstrator's intended actions. *CADSS* and *SLBAS* (lines 4–5) come from noisy examples and since their frequencies in training data are very low, they are assigned much lower probabilities. (b) A sample grammar learned from an ideal (noise-free) dataset. (c) A sample grammar learned from the same dataset of (a), but with the pruning threshold of 0.15. Please see Section 5.2.3 for more detailed analysis on pruning effects.

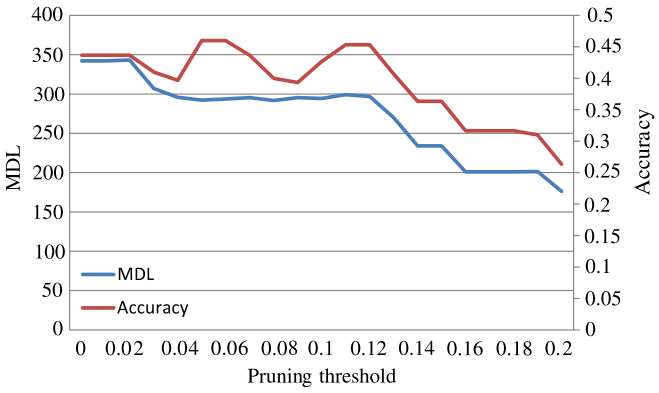


Fig. 13. The effect of different pruning parameters. In this experiment, we trained from all training data, i.e. all samples from both low-noise and high-noise conditions, and similarly tested on all testing samples. It can be seen that although overall MDL score decreases as threshold increases, the resulting grammar loses generality and shows poor performance on testing data. As in the previous experiments, a trial is regarded as fail even if there was a single error in parsed symbols.

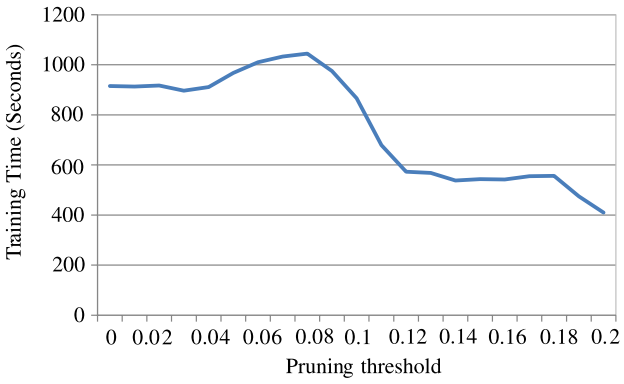


Fig. 14. The comparison of training times over different prune parameters. Since rules are more likely to be pruned as the threshold increases, the overall learning time tends to decrease. It was tested on a Linux desktop with i7 3.2 GHz, 16 GB RAM, Python 3.2.

symbols in the input sequences, the underlying structures of action components are successfully captured. The knowledge of these underlying structures allow to filter out contextually inconsistent observations. For example, the learned action component *LAD* allows the action DROP (*D*) to be expected when MOVE BETWEEN (*A*) action is observed, even if DROP action was missed or misinterpreted. The last line of the grammar rules shows that it also captured the recursiveness nature of the task.

Although each model is constructed only from 6 sample sequences, it successfully captured these core components due to the active substring searching explained in Section 4.1. Fig. 12(b) shows an example grammar constructed from data that contains no noise. Most of the experiments, however, include noise symbols in the input sequence which hinders the discovery of the full meaningful action component such as *LADLBDLCD* in Fig. 12(b), line 1. Nevertheless, grammars discovered like the one in Fig. 12(a) worked reasonably well to support parsing the same task with more complicated sequences.

5.2.3. Analysis on different pruning factors

In this section, we show how the change of pruning thresholds affect the result. The range of thresholds are 0.00–0.20, with intervals of 0.01. Figs. 13 and 14 show how the pruning threshold affects the testing accuracy and grammar induction time. In this experiment, we train from all training data, i.e. all samples from both low-noise and high-noise conditions, and test on all testing samples as well. As in the previous experiments, a trial is regarded as fail even if there was a single error in parsed symbols.

The result in Fig. 13 shows that although overall MDL score decreases as the threshold increases, the resulting grammar loses generality and shows poor performance on testing data. In Fig. 14, training time generally decreases as the pruning threshold increases, as more rules are more likely to be discarded during the induction process.

5.3. Dance imitation learning

In this experiment, we define 3 types of dance demonstrations. The goal of this experiment is to learn the generalized representation of human dance movements, which is utilized to recognize more complex movements. Each dance sequence is composed of a subset of predefined motion primitives, i.e. dance symbols.

The inputs to the system are time-series 54-dimensional angular values of 18 human joints, captured using an OptiTrack 8-camera motion capture system. Temporal segmentation is applied (Section 5.3.1), where each segment is mapped to one of 9 primitive dance symbols. To map segments to probabilistic symbols, we need to train detectors (Section 5.3.2). After obtaining detectors, we can now convert a video stream into a sequence of symbols which is fed into our SCFG learning framework. Finally, the robot performs the dance by executing the parsed symbols. We map the human joints into iCub's joints and generalize the trajectories of 9 motion primitives from multiple demonstrations (Section 5.3.3).

The 3 dance grammars used to generate actions are: (1) $(CD)^n$ $(EF)^n$, (2) $(ABE)^n$, and (3) (H^nGI^n). We describe the scenario settings in Fig. 16.

5.3.1. Temporal segmentation

We modify the temporal segmentation method proposed by Fod et al. [42] which segments human motions at zero-crossing points of the squared sum of joint velocities.

Similar to [42], where they selected a subset of joints, we select four sets of human joints (usually between 3 and 5 out of 18) that move significantly in every motion primitive, as shown in Fig. 17. We then compute two types of features for segmentation: the average of squares of joint velocity (ASV, Eq. (11)) and the average of squares of joint distance to the initial posture of the dance sequence (ASD, Eq. (12)). An example is shown in Fig. 18.

$$ASV(\mathcal{S}, \omega) = \sum_{i \in S} \omega_i^2 / Card(\mathcal{S}) \tag{11}$$

$$ASD(\mathcal{S}, \theta, \theta^r) = \sum_{i \in S} (\theta_i - \theta_i^r)^2 / Card(\mathcal{S}) \tag{12}$$

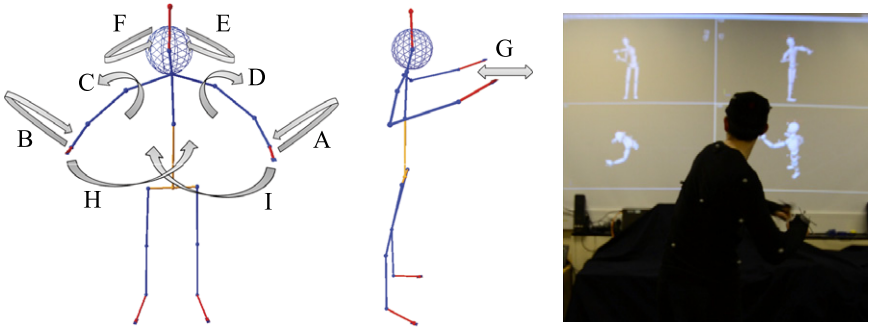


Fig. 15. 9 motion primitives used in this experiment and a demonstration example. Please see the following video for better visualization: http://www.youtube.com/watch?v=S99ViThK050.

Grammar	$(CD)^n(EF)^n$	$(ABE)^n$	H^nGI^n
Train set	n=1,2	n=1,2,3	n=1,2,3
Test set	n=3,4	n=4,5	n=4,5

Fig. 16. 3 types of dance representations used in the experiment. Please see Fig. 15 for reference. In the training set, there are 5 trials for each value of n (sequence length), which result in 40 dance demonstrations. (Total of 225 input symbols). The testing set has 6 trials for each n, which result in 36 dance demonstrations. (Total of 450 input symbols).

where $\mathcal{S}$ is the set of joints as defined in Fig. 17, ω_i is the velocity of joint i, $Card(\mathcal{S})$ is the cardinal number of $\mathcal{S}$, θ_i is the position of joint i, and θ^r is the vector of joints position of the reference posture.

For each time step, we choose $\mathcal{S}$ with the largest ASV value for segmentation. Then we find the zero crossings of the ASV where ASD value is lower than a threshold. In our case, the threshold is automatically computed from the data by clustering ASD values into two groups and taking the mean of two cluster centers. We use K-means ($K = 2$) for clustering. An example is shown in Fig. 18.

5.3.2. Training of symbol detectors

After obtaining video segments, we first compute the angular velocity of joints by computing the frame differences of 54-dimensional joint data, followed by taking Bag-of-Words (BoW) approach combined with one-vs-all SVM. We cluster the joint velocity data into K clusters using K-means ($K = 50$), and use them to compute the histogram of every segment. One-vs-all multiclass SVM classifiers are trained from these histograms for 9 different symbols using radial basis function (RBF) kernel. We use LibSVM library [43] to train and test SVMs. After running a grid search optimization, we obtained RBF kernel parameters of $C = 0.5$, $\gamma = 0.0078125$.

5.3.3. Generalizing trajectories

After classifying each segmentation, we use all the segments that belong to the same class as the training set to generalize the trajectories for iCub. Dynamic Time Warping [44] is applied to demonstration sets to gain trajectories for each motion primitive, which are then mapped to the corresponding joints of iCub. The joint configurations of iCub's chest and head are the same as those of human, which makes it possible to directly assign the angles of these joints to iCub. But the configurations of iCub's arm and the human arm are different, so we map these joint angles to the iCub by minimizing the error of the directional vectors of the upper and the lower arm between the human and iCub under the constrains of the joint limits of iCub's arm. Now iCub is ready to execute the sequence of dance symbols. Fig. 19 shows the representative frames of one of 3 dance sequences.

5.3.4. Results and analysis

Fig. 20 shows the performance in 4 scenarios, similar to the Towers of Hanoi experiment in Section 5.2. We denote as the low-noise condition (L) when the ground-truth segmentation is used, and the high-noise condition (H) when automatic segmentation described in Section 5.3.1 is used. The first letter corresponds to the training condition, whereas the second letter corresponds to the testing condition. For example, "LH" means the grammar was learned using manually segmented sequences from the training dataset, and parsed on automatically segmented sequences from the testing dataset. Since there are a significant number of input error symbols, we also denote the actual number of symbols that are recognized correctly for comparison. In the pure imitation (no grammar) case, the number of correct symbols is equivalent to the number of correctly recognized symbols by symbol detectors.

Fig. 21 shows the learned grammars of 3 dance representations from the demonstrations using automatically segmented sequences as training dataset computed by the method described in Section 5.3.1. This training dataset is marked as the high-noise case (H) since the higher error in the segmentation generally leads to a higher error rate on the symbol detection, which affects on grammar learning. Thus, these grammars are used to test scenarios "HL" and "HH".

Fig. 22 shows the learned grammars using manually segmented sequences as training dataset. It is notable that only the segmentation part was done manually. The training and testing of symbol detectors and grammar learning parts are all done in the same way as in the automatically segmented dataset. These grammars are used to test scenarios "LL" and "LH".

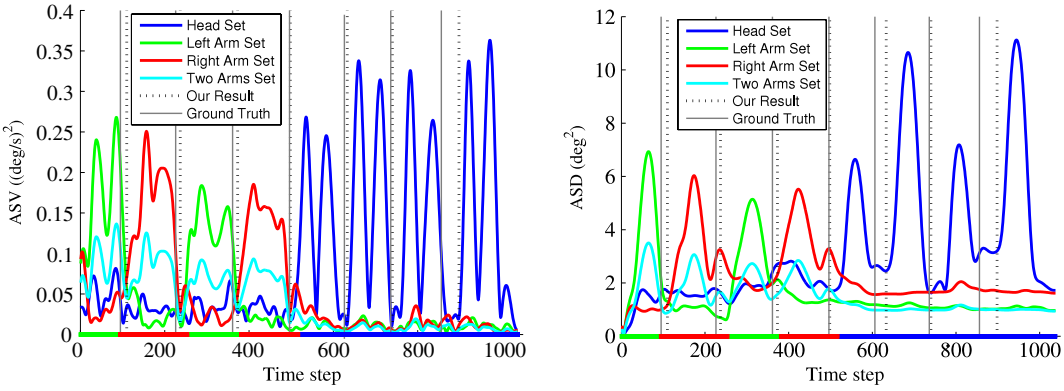
In Fig. 20 (HL scenario), it can be seen that the pure imitation has a better performance than using grammars obtained using the baseline method. This is because of the high level of noise in the input hinders building a correct representation in the grammar. As a result, it sometimes leads to an adverse effect where the correct input symbols are identified as wrong. Our proposed method is less likely to suffer from this problem because the uncertainty values of input symbols and substring frequencies are considered.

The grammars shown in Fig. 21(b) (c) and Fig. 22(a) (b), actually captured the original grammar used to generate dance sequences, although the last one contains some unrelated symbols due to the higher level of symbol detector errors. They can effectively correct the wrong symbol patterns that largely differ from the symbol patterns in training sequences. Still, it is interesting to see that other two grammars partially capture the important constraints such as "HSI" and "HG" in Fig. 21(c) and "EF" and "CD" in Fig. 22(a).

For the execution of motion primitives, we concatenate learned trajectories of joints based on parsed symbol sequence and apply a low-pass filter to avoid jerky movements due to discontinuity between symbols. Since all trajectories are learned from multiple human demonstrations, iCub exhibits natural human-like movements, such as subtle movements of torso and head while reaching an arm forward. A video of a demonstrator example can be found at: http://www.youtube.com/watch?v=S99ViThK050.

Joints Set	Involved Human Joints	Motion Primitives
Left arm	Chest, Left shoulder & Elbow	A, D, I
Right arm	Chest, Right shoulder & Elbow	B, C, H
Two arms	Chest, Left and Right shoulder & Elbow	G
Head	Chest, Neck, Head	E, F

Fig. 17. The informative human joints chosen to be used for calculating the ASV and ASD values. As these joints are often overlapped across multiple motion primitives, the number of the joint sets are reduced to four.



(a) ASV and segmentation of dance CDCDEFEF.

(b) ASD and segmentation of dance CDCDEFEF.

Fig. 18. The ASV (a) and ASD (b) of the movement sequence: the used joints set for each time step is marked on the bottom using corresponding color. The zero-crossings of ASV with sufficiently low ASD value are chosen as the segmentation points.

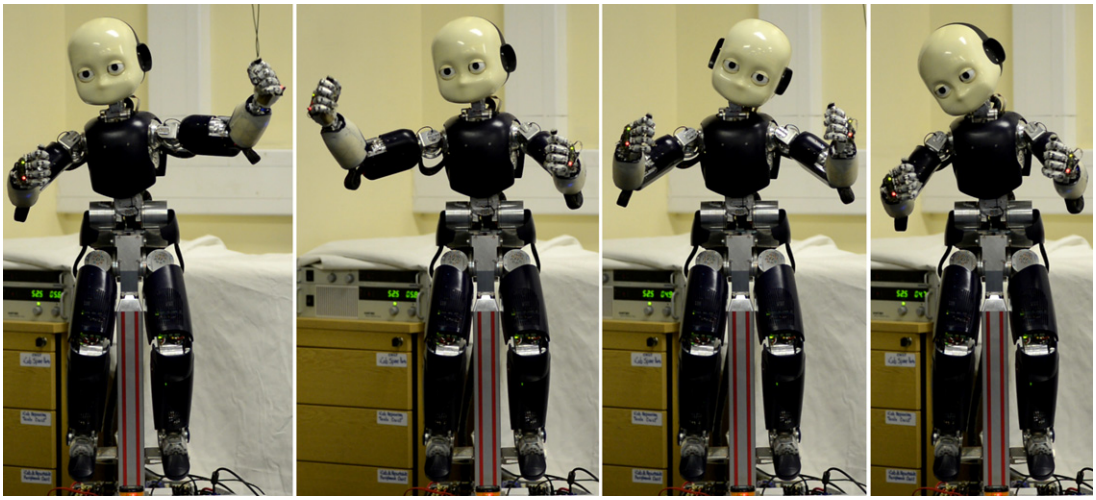


Fig. 19. iCub performing parsed actions. Each figure from the left to right respectively represents actions C, D, E, and F. The full movement video can be seen on: http://www.youtube.com/watch?v=S99ViThK050.

Scenario	Method	# Correct	Success	Avg.MDL	Scenario	Method	# Correct	Success	Avg.MDL
LL	Proposed	450	36	400.09	LH	Proposed	450	35	413.63
	Base	450	36	408.70		Base	450	35	422.26
	Pure Imi.	437	30	N/A		Pure Imi.	347	11	N/A
HL	Proposed	450	36	450.99	HH	Proposed	424	30	464.27
	Base	414	24	464.93		Base	414	24	476.12
	Pure Imi.	437	30	N/A		Pure Imi.	347	11	N/A

Fig. 20. Detailed results with average MDL scores for comparison. Each scenario has 36 sequences, and the total number of symbols per scenario is 450. "# Correct" shows the number of correctly recognized symbols after parsing, where the pure imitation (no activity grammar learning) case shows the raw error symbols (detector output). MDL score is not available for the pure imitation as it does not rely on any learned model. It can be seen that the lower MDL scores generally lead to higher success rates.

6. Discussions and future directions

We have presented a robot imitation learning framework using probabilistic activity grammars. Our method aims to discover reusable common action components across multiple tasks from input stream. We have shown in the two non-trivial real-world experiments (Sections 5.2 and 5.3) that our method is capable to learn reusable structures under reasonable amount of noise, in

a
S→EF [0.293200]
| SS [0.287079]
| CD [0.198005]
| SSSS [0.085637]
| CF [0.044922]
| CES [0.028048]
| CHS [0.024241]
| SFE [0.019778]
| SCIHFS [0.019089]

b
S→ABE [0.592059]
| SS [0.390003]
| SAAB [0.017939]

c
S→HGI [0.523234]
| HSI [0.415843]
| HESII [0.034387]
| HSG [0.026536]

Fig. 21. Acquired grammars from automatically segmented dataset using the method described in Section 5.3.1. The error in the segmentation leads to a higher error rate on detectors, which is regarded as the high-noise scenario.

a
S→CDEF [0.667192]
| CDSEF [0.332808]

b
S→ABE [0.598758]
| SS [0.401242]

c
S→HS [0.307153]
| SI [0.259863]
| HSI [0.257960]
| HG [0.144169]
| SG [0.020607]
| SF [0.010248]

Fig. 22. Learned grammars from manually segmented dataset, noted as the low-noise scenario. Note that only segmentation was done manually, where symbol detectors are still trained and tested in the same way as in automatically-segmented dataset.

addition to the synthetic dataset experiment for systematic analysis. In the Dance Imitation experiment (Section 5.3). The robot not only generalized the task from multiple demonstrations at the symbolic level, but also at the trajectory level, which makes our framework more complete. We have also experimentally shown that a lower MDL score generally leads to higher performance on parsing unforeseen action sequences.

The discovery of important component actions and recursions was critical to the performance, which is supported by the results reported in Sections 5.2.2 and 5.3.4. For example, the action component *LAD* in Fig. 12(a), line 1 (Lift a disk, Move between towers 1 and 2, Drop) provides local constraints that enforce contextually consistent interpretation by biasing the parser to parse in the order of *L–A–D* even when the observed symbols are partially wrong. This biasing effect can be also interpreted as an affordance learning, similar to [45], where the recognition of an observed gesture depends on a context variable. Using the learned grammar in Fig. 22(a), when the robot observes *CD* actions several times, it can "anticipate" the same number of *EF* actions, which acts as a belief system. Due to this advantage, wrong or uncertain symbols were often corrected or clarified by the learned grammar, e.g. Fig. 11(a) and (b). This action anticipation could be a useful tool for active learning.

The results reported in Section 5.1 support our idea that handling uncertainty values of input symbols improves the performance. Also, the human-readable results, e.g. Figs. 12, 21 and 22, is another benefit point in human–robot interaction domain, which often involves non-experts.

We have shown in Section 5.2.3 how the pruning threshold affects the overall performance. Although we have used a fixed pruning factor for all experiments, it would be an interesting work to find an optimal parameter in a more systematic way, e.g. cross-validation within training samples. This will lead to a more compact representation of activities while keeping the training time to minimum.

In the *Bag of Balls* (Section 5.1) and *The Towers of Hanoi* (Section 5.2) experiments, we have used 3 and 5 primitive symbols, respectively. While these were simple enough to explain how the proposed method works, our *Dance* experiment scales up to 9 primitive symbols, which are similar to other real-world settings, e.g. 10 primitive symbols used to model complex employee–customer transaction activities in a convenience store [29], 10 primitive symbols used to model car–human interaction scenarios in surveillance videos [22], and 12 primitive symbols to model the Black Jack card game [46]. The scalability of these methods to even more complex datasets that will necessitate even higher number of symbols remains an open challenge.

In *the Towers of Hanoi* experiment, we had an assumption where the robot knows that only a smaller object should be placed over larger object, similar to how humans tell others when they give instructions on solving this specific puzzle. However, it would be an interesting work to make this more general, by making a robot to learn this rule well by using symbolic-level planners, such as STRIPS-like symbolic planners [47].

The inclusion of domain-dependent, biased structural priors could be also beneficial in terms of both searching speed and grammar accuracy as certain models will be effectively rejected even if they retain good MDL scores. This will be especially useful in the domain of imitation learning which often shares many reusable components across different tasks.

Acknowledgments

This work was supported by the EU FP7 projects EFAA (FP7-ICT-270490) and ALIZ-E (FP7-ICT-248116).

References

[1] A. Billard, S. Calinon, R. Dillmann, S. Schaal, Robot Programming by Demonstration (Chapter 59), Springer, 2008.
[2] Y. Kuniyoshi, M. Inaba, H. Inoue, Learning by watching: extracting reusable task knowledge from visual observation of human performance, T. Robotics and Automation 10 (1994) 799–822.
[3] R. Dillmann, Teaching and learning of robot tasks via observation of human performance, Robotics and Autonomous Systems 47 (2–3) (2004) 109–116.
[4] S. Schaal, Is imitation learning the route to humanoid robots? Trends in Cognitive Sciences 3 (6) (1999) 233–242.
[5] M. Asada, M. Ogino, S. Matsuyama, J. Ooga, Imitation learning based on visuo-somatic mapping, Experimental Robotics IX (2006) 269–278.
[6] K. Dautenhahn, C. Nehaniv, The agent-based perspective on imitation, in: Imitation in Animals and Artifacts, 2002, pp. 1–40.
[7] J. Demiris, G. Hayes, Imitation as a dual-route process featuring predictive and learning components; a biologically plausible computational model, in: K. Dautenhahn, C. Nehaniv (Eds.), Imitation in Animals and Artifacts (Chapter 13), 2002, pp. 327–361.
[8] S. Calinon, F. Guenter, A. Billard, Goal-directed imitation in a humanoid robot, in: IEEE International Conference on Robotics and Automation, 2005, pp. 299–304.
[9] D.C. Bentivegna, C.G. Atkeson, G. Cheng, Learning similar tasks from observation and practice, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006, pp. 2677–2683.
[10] K. Lee, J. Lee, A. Thomaz, A. Bobick, Effective robot task learning by focusing on task-relevant objects, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, USA, 2009, pp. 2551–2556.
[11] C. Chao, M. Cakmak, A.L. Thomaz, Towards grounding concepts for transfer in goal learning from demonstration, in: IEEE International Conference on Development and Learning, Vol. 2, 2011, pp. 1–6.

[12] D. Baldwin, J. Baird, Discerning intentions in dynamic human action, Trends in Cognitive Sciences 5 (4) (2001) 171–178.
[13] A. Woodward, J. Sommerville, J. Guajardo, How infants make sense of intentional action, in: Intentions and Intentionality: Foundations of Social Cognition, 2001, pp. 149–169.
[14] A. Billard, S. Calinon, F. Guenter, Discriminative and adaptive imitation in uni-manual and bi-manual tasks, Robotics and Autonomous Systems 54 (5) (2006) 370–384.
[15] T. Asfour, P. Azad, F. Gyarfas, R. Dillmann, Imitation learning of dual-arm manipulation tasks in humanoid robots, International Journal of Humanoid Robotics 5 (2) (2008) 183–202.
[16] Y. Wu, Y. Demiris, Towards one shot learning by imitation for humanoid robots, in: IEEE International Conference on Robotics and Automation, 2010, pp. 2889–2894.
[17] D. Nguyen-tuong, J. Peters, Local gaussian process regression for real time online model learning and control, in: Advances in Neural Information Processing Systems, 2008, pp. 1193–1200.
[18] S. Gurbuz, T. Shimizu, G. Cheng, Real-time stereo facial feature tracking: mimicking human mouth movement on a humanoid robot head, in: IEEE-RAS International Conference on Humanoid Robots, 2005, pp. 363–368.
[19] H. Soh, Y. Su, Y. Demiris, Online spatio-temporal Gaussian process experts with application to tactile classification, IEEE/RSJ International Conference on Intelligent Robots and Systems (2012) 4489–4496.
[20] Y. Demiris, B. Khadhouri, Hierarchical attentive multiple models for execution and recognition of actions, Robotics and Autonomous Systems 54 (5) (2006) 361–369.
[21] M. Ryoo, J. Aggarwal, Robust human–computer interaction system guiding a user by providing feedback, in: International Joint Conferences on Artificial Intelligence, 2007, pp. 2850–2855.
[22] Y. Ivanov, A. Bobick, Recognition of visual activities and interactions by stochastic parsing, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (2000) 852–872.
[23] K. Lee, Y. Demiris, Towards incremental learning of task-dependent action sequences using probabilistic parsing, in: IEEE International Conference on Development and Learning, Vol. 2, Frankfurt, Germany, 2011, pp. 1–6.
[24] C. de la Higuera, A bibliographical study of grammatical inference, Pattern Recognition 38 (9) (2005) 1332–1348.
[25] Y. Aloimonos, G. Guerra-Filho, A. Ogale, The language of action: a new tool for human-centric interfaces, in: H. Aghajan, J. Augusto, R. Delgado (Eds.), Human Centric Interfaces for Ambient Intelligence, 2009, pp. 95–131.
[26] C. Nevill-Manning, I. Witten, Identifying hierarchical structure in sequences: a linear-time algorithm, Journal of Artificial Intelligence Research 7 (1997) 67–82.
[27] Z. Solan, D. Horn, E. Ruppin, S. Edelman, Unsupervised learning of natural languages, Proceedings of the National Academy of Sciences 102 (33) (2005) 11629–11634.
[28] A. Stolcke, S. Omohundro, Inducing probabilistic grammars by Bayesian model merging, Grammatical Inference and Applications 862 (1994) 106–118.
[29] K. Kitani, S. Yoichi, A. Sugimoto, Recovering the basic structure of human activities from noisy video-based symbol strings, International Journal of Pattern Recognition and Artificial Intelligence 22 (08) (2008) 1621–1646.
[30] K. Lee, T.K. Kim, Y. Demiris, Learning action symbols for hierarchical grammar induction, in: The 21st International Conference on Pattern Recognition, Tsukuba Science City, Japan, 2012, pp. 3778–3782.
[31] A. Ogale, A. Karapurkar, Y. Aloimonos, View-invariant modeling and recognition of human actions using grammars, Dynamical Vision (2007) 115–126.
[32] M. Nicolescu, M. Mataric, Natural methods for robot task learning: instructive demonstrations, generalization and practice, in: International Joint Conference on Autonomous Agents and Multiagent Systems, 2003, pp. 241–248.
[33] P. Langley, S. Stromsten, Learning context-free grammars with a simplicity bias, in: The European Conference on Machine Learning, Vol. 1810, 2000, pp. 220–228.
[34] C. Nevill-Manning, I. Witten, On-line and off-line heuristics for inferring hierarchies of repetitions in sequences, Proceedings of the IEEE 88 (11) (2002) 1745–1755.
[35] F. Zhou, F. Torre, J. Hodgins, Aligned cluster analysis for temporal segmentation of human motion, in: IEEE International Conference on Automatic Face & Gesture Recognition, 2008, pp. 1–7.
[36] Y. Sakakibara, M. Brown, R. Hughey, S. Mian, K. Sjölander, R.C. Underwood, D. Haussler, Recent methods for RNA modeling using stochastic context-free grammars, in: Combinatorial Pattern Matching, Springer, 1994, pp. 289–306.
[37] Y. Shan, R.I. McKay, R. Baxter, H. Abbass, D. Essam, H. Nguyen, Grammar model-based program evolution, in: Congress on Evolutionary Computation 2004, Vol. 1, CEC2004, IEEE, 2004, pp. 478–485.
[38] G. Metta, G. Sandini, D. Vernon, L. Natale, F. Nori, The iCub humanoid robot: an open platform for research in embodied cognition, in: Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems, ACM, 2008, pp. 50–56.
[39] G. Bradski, The Opencv library, Doctor Dobbs Journal 25 (11) (2000) 120–126.
[40] U. Pattacini, F. Nori, L. Natale, G. Metta, G. Sandini, An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010, pp. 1668–1674.
[41] Y. Su, Y. Wu, K. Lee, Z. Du, Y. Demiris, Robust grasping for an under-actuated anthropomorphic hand under object position uncertainty, in: IEEE-RAS International Conference on Humanoid Robots, Osaka, Japan, 2012, pp. 719–725.
[42] A. Fod, M. Mataric, O. Jenkins, Automated derivation of primitives for movement classification, Autonomous Robots 12 (1) (2002) 39–54.
[43] C. Chang, C. Lin, LIBSVM: a library for support vector machines, ACM Transactions on Intelligent Systems and Technology 2 (3) (2011) 1–27.
[44] C.-Y. Chiu, S.-P. Chao, M.-Y. Wu, S.-N. Yang, H.-C. Lin, Content-based retrieval for human motion data, Journal of Visual Communication and Image Representation 15 (3) (2004) 446–466.
[45] M. Lopes, J. Santos-Victor, Visual learning by imitation with motor representations, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 35 (3) (2005) 438–449.
[46] D. Moore, I. Essa, Recognizing multitasked activities from video using stochastic context-free grammar, in: AAAI/IAAI, 2002, pp. 770–776.
[47] R.E. Fikes, N.J. Nilsson, STRIPS: a new approach to the application of theorem proving to problem solving, Artificial Intelligence 2 (3) (1972) 189–208.

Kyuhwa Lee is a Ph.D. student and a research assistant in the Personal Robotics Lab at Imperial College London. His research interests focus upon structured human task learning and active learning for robots using syntactic approaches. He actively works in the fields of robot learning by demonstration with the real world applications on humanoid robots such as iCub and Simon.

Yanyu Su is a Ph.D. student at the State Key Laboratory of Robotics and System at Harbin Institute of Technology, and is currently visiting the Personal Robotics Library at Imperial College London working on biomimetic grasping mechanisms for complex humanoid hands, and robot learning by demonstration.

Tae-Kyun Kim is a Lecturer in the computer vision and learning at the Imperial College London, UK, since 2010. He obtained his Ph.D. from University of Cambridge in 2007 and was a research fellow of Sidney Sussex College in Cambridge during 2007–2010. His research interests span various topics including: object recognition, tracking, face recognition and surveillance, action/gesture recognition and semantic image segmentation and reconstruction. He has co-authored over 40 journal and conference papers, 6 MPEG7 standard documents and 17 international patents. His co-authored algorithm is an international standard of MPEG-7 ISO/IEC for face image retrieval.

Yiannis Demiris is a Reader of Imperial College London. His research interests include assistive robotics, multi-robot systems, robot human interaction and learning by demonstration. Dr. Demiris' research is funded by the UK's Engineering and Physical Sciences Research Council (EPSRC), the Royal Society, BAE Systems, and the EU FP7 program through projects ALIZ-E and EFAA, both addressing novel machine learning approaches to human–robot interaction. Dr. Yiannis Demiris has guest edited special issues of the IEEE Transactions on SMC-B specifically on Learning by Observation, Demonstration, and Imitation, and of the Adaptive Behavior Journal on Developmental Robotics. He has organized six international workshops on Robot Learning, BioInspired Machine Learning, Epigenetic Robotics, and Imitation in Animals and Artifacts (AISB), was the chair of the IEEE International Conference on Development and Learning (ICDL) for 2007, as well as the program chair of the ACM/IEEE International Conference on Human–Robot Interaction (HRI) 2008. in 2012 he received the Rector's award for Teaching Excellence, and the Faculty of Engineering Award for Excellence in Engineering Education. He is a Senior Member of IEEE, and a member of the Institute of Engineering & Technology of Britain (IET).