Learning Reusable Task Components using Hierarchical Activity Grammars with Uncertainties

Kyuhwa Lee, Tae-Kyun Kim and Yiannis Demiris

*Abstract***—**

We present a novel learning method using activity grammars capable of learning reusable task components from a reasonably small number of samples under noisy conditions. Our linguistic approach aims to extract the hierarchical structure of activities which can be recursively applied to help recognize unforeseen, more complicated tasks that share the same underlying structures. To achieve this goal, our method 1) actively searches for frequently occurring action symbols that are subset of input samples to effectively discover the hierarchy, and 2) explicitly takes into account the uncertainty values associated with input symbols due to the noise inherent in low-level detectors. In addition to experimenting with a synthetic dataset to systematically analyze the algorithm's performance, we apply our method in human-led imitation learning environment where a robot learns reusable components of the task from short demonstrations to correctly imitate more complicated, longer demonstrations of the same task category. The results suggest that under reasonable amount of noise, our method is capable to capture the reusable structures of tasks and generalize to cope with recursions.

I. Introduction

Humans have a natural ability to learn new activity representations despite noisy sensory inputs by using previously learned action components, since many human activities are inherently structured. This ability can be also found in the process of language acquisition, where a child acquires more sophisticated concepts by incorporating previously learned vocabularies.

Motivated by this observation, we are interested in learning reusable action components to better understand more complicated tasks that share the same structures under noisy environments. Stochastic context-free grammars (SCFGs) are used as our framework since 1) they are robust to noise due to their probabilistic nature, 2) they provide a compact way to represent hierarchical and recursive structures, and 3) they output human-readable results which can be interpreted intuitively for non-experts. Other commonly used techniques such as HMMs provide faster recognition speed but they are less expressive as they are based on regular grammars. For example, the following language cannot be represented: $a^n b^n$, where a=Push, b=Pull (equal number of Push and Pull operations.)

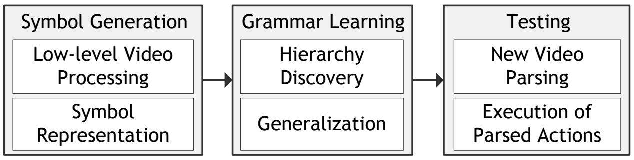

Fig. 1. Overview of the framework.

A large amount of effort has been spent to understand tasks using context-free grammars(CFGs). In [1], 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 [2], Ivanov defines SCFG rules to recognize more complicated actions, e.g. music conducting gestures, using HMM-based low-level action detectors. In [3], a robot imitates human demonstrations of organizing objects using SCFG-based task-independent action sequences. There are also several other areas that utilize CFGs as the framework such as computational biology and speech recognition, as mentioned in [4]. Aloimonos et al. [5] give detailed explanations about various useful applications that use linguistic approach including human motoric action representations.

The aforementioned studies consider cases where the proper grammar rules are given in prior. As opposed to manually defining the grammar rules to represent a task, there are also several works aiming to construct (i.e. induce) grammars from data. In early work, Nevill-Manning et al. [6] presented the SEQUITUR algorithm which can discover hierarchy among symbols. Solan et al.[7] presented the ADIOS algorithm which induces CFGs and Context-sensitive grammars as well with some restrictions (e.g. no recursions) using graphical representations. Stolcke and Omohundro [8] presented a SCFG induction technique, which more recently has been extended by Kitani et al. [9] to remove task-irrelevant noisy symbols to cope with more realistic environments. More detailed reviews will be given in Sec. II.

Compared with the conventional learning techniques, our method has two distinctive features: 1) Our method actively searches for frequently occurring sub-strings from the input stream that are likely to be meaningful to discover hierarchical structures of activity. 2) We take into account the uncertainty values of the input symbols computed by low-level atomic action detectors. Similar to Ivanov's work [2] 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.

All authors are with the Intelligent Systems and Networks group, Department of Electrical and Electronic Engineering, Imperial College London, UK. {k.lee09, tk.kim, y.demiris}@imperial.ac.uk

In this paper we study how learning activity grammars can be learned from human partners. We assume that 1) the system can detect meaningful atomic actions which are not necessarily noise-free, and 2) extensive complete data sets are not always available but numerous examples of smaller component elements could be found.

II. Related Work

Our framework (Fig. 1) shares the concept of imitation learning presented in [10], [11], where a robot learns a new task directly from human demonstration without the need for extensive reprogramming. We are inspired by the work in [12] which shares the same motivation of hierarchical learning with our work. 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 [8], 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 by recursive rules. 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 meaningful and do not contain any certainty 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 n-gram-like frequency table within each input sample. This allows us to generalize the rule more effectively with the same amount of data samples that better represents data.

More recently, Kitani et al. [9] presented a framework of discovering human activities from video sequences using SCFG induction technique based on [8]. By assuming that the noise symbols are not 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, because noise symbols are not assumed to be part of task representation, it is limited to dealing with insertion errors where wrong symbols are accidentally inserted.

III. Background

A. 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 productions rules, and S is the start symbol. The production rules take the form $X \to \lambda$, 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 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 "best" grammar is found. The quality of a grammar is measured by the Minimum Description Length (MDL) principle as used in [13][9][8], which will be explained more in Sec. III-B.

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. 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.

B. Measuring the Quality of a Grammar

The general objective is to find a grammar that is sufficiently simple yet expressive as pointed out by Langley et al. [13] We denote $P(M)$ as *a priori* model probability, where M includes only the structure and parameter priors which does not consider the input data D, and $P(D|M)$ as the posterior probability given the model. Then, our goal is to minimize the MDL score as -log of joint probability

$$-logP(M, D) = -logP(M) - logP(D|M) \qquad (1)$$

The posterior probability $P(D|M)$ is computed using Viterbi parsing, which is commonly used in HMMs. However, unlike [8] and [9], to handle the uncertainty values of the input symbols, the method of computing the posterior probability needs to be modified. To cope with this situation, we use the SCFG parsing algorithm with uncertainty input introduced in [2] to compute the posterior description length.

IV. Proposed Method

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

A. Active Substring Discovery

To generate a grammar that focuses on patterns with strong regularity, we build an n-gram-like frequency table such that it keeps the number of occurrences of substrings that are subset of input sequences. The score of a rule $X \to \lambda$ is the occurrence value of λ in the frequency table multiplied by the expected probability value of λ. Its calculation will be discussed in the following section, Sec. IV-B.

For simplicity, we first consider the case without uncertainty values. In this case, as defined in [8] and [9], the rule probability is calculated by normalizing rule scores, i.e.:

$$P(X \to \lambda_i) = \frac{f(X \to \lambda_i)}{\sum_k f(X \to \lambda_k)} \qquad (2)$$

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

$$\sum_i P(X \to \lambda_i) = 1 \qquad (3)$$

(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]
X→AB (27) [0.96]
| AC (1) [0.04]
Z→XX (13) [1.00]

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

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

Fig. 2. (a) Initial naive grammar (b) After $Substituting$ AB with X, AC with Y, and XX with Z (c) After $Merging$ (X, Y) to X (d) After $Mergeing$ (X, Z) to Z (e) After $Merging$ (S, Z) to S

Unlike [8] and [9], the symbol counts are computed directly from the occurrences of each input sequence as whole, not subset. However, 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 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, respectively. We now apply a *Substitute* (Fig. 2(b)) and *Merge* operators (Fig. 2(c)-(e)) introduced in [8] with rule scores obtained from our frequency table. We have 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. This "soft" classification is one of the advantages of SCFGs.

In practice, it is often useful to limit the maximum length of symbols to be considered in frequency table to avoid generating exhaustive list of symbols to increase the speed. This is reasonable assumption as human activities usually involve repetitive action components[14]. Since the search space of the possible grammars is not small, a beam search strategy is applied as in [8] 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.

B. Considering Input Samples with Uncertainty

So far, we have only considered a case where input symbols are non-probabilistic, i.e. terminals $(a, b, c...)$ 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 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 $\lambda = s_1 s_2 s_3 ... s_n$ of length n from input, as

$$P(\lambda) = (\prod_n P(s_n))^{\frac{1}{n}} \qquad (4)$$

The term $\frac{1}{n}$ is used to normalize the probability as the probability will always decrease as λ gets lengthier. The expected value of λ is obtained by averaging all occurrences of λ in the input. Thus, we modify the equation (2) as

$$P(X \to \lambda_i) = \frac{f(X \to \lambda_i)\mu(\lambda_i)}{\sum_k f(X \to \lambda_k)\mu(\lambda_k)} \qquad (5)$$

where $\mu(\cdot)$ denotes the expected value and i denotes the i-th rule of X. We use this equation throughout our experiments.

In our method, we define the model prior probability

$$P(M) = P(M_S, M_\theta) = P(M_S)P(M_\theta|M_S) \qquad (6)$$

where $P(M_S)$ denotes structure prior and $P(M_\theta)$ denotes parameter prior. As in [8] and [9], $P(M_S)$ is defined as Poisson distribution with mean (average production length) 3.0. $P(M_\theta|M_S)$ is defined as the product of Dirichlet distributions, such that each Dirichlet distribution represents uniformly distributed probability across all possible productions of a non-terminal symbol X, i.e.:

$$P_X(M_\theta|M_S) = \frac{1}{\beta(\alpha_1, ..., \alpha_n)} \prod_{i=1}^{n} {\theta_i}^{\alpha_i - 1}, \qquad (7)$$

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

V. Experiments and Analysis

To test our framework, we first experiment on synthetic data with systematically varying the levels of noise, followed by real-world data obtained from 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.

We apply pruning process as in [8] to speed up the induction and filter out non-critical production rules having probabilities lower than some threshold τ, as they are often accidentally created due to noise. If the removal of a rule leads to a decrease in the description length of model prior but increases the posterior description length in relatively small amount, it will lead to a better (lower) MDL score. We set $\tau = 0.01$ in all of our experiments.

A. Bag-of-Balls Experiment

In this experiment, we assume a scenario where arbitrary number of balls are inserted into a bag (denoted as a), moved to another place (denoted as b), and the same number of balls are taken out later (denoted as c), which can be represented in the form $a^n b c^n$. 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

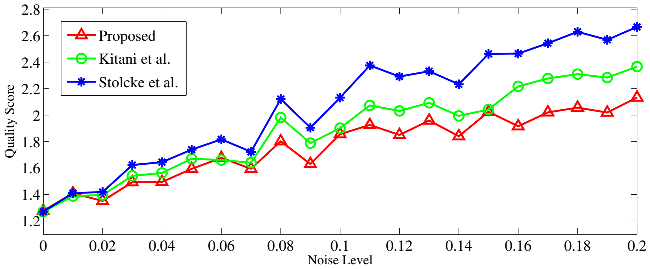

Fig. 3. Quality scores of grammars generated by various methods. The lower score indicates that the grammar is more compact yet maintains enough expressive power.

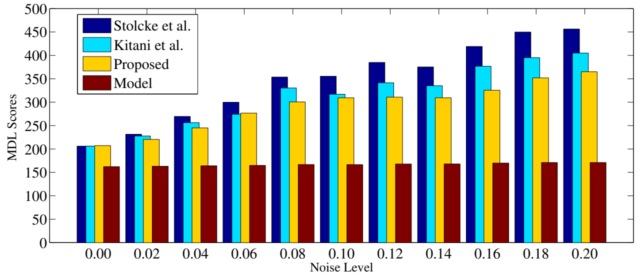

Fig. 4. Actual MDL scores for each method compared with the model grammar. MDL scores are averaged over 10 trials for each noise condition. The graph is shown with 2% step for better view. The lower score indicates that the grammar is more compact yet reasonably expressive. How these scores affect the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

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 conduct each experiment with two previously reviewed methods reported by Kitani [9] and Stolcke [8].

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 [9].

The quality score 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 quality scores over various noise conditions, where in most cases the grammars generated by our proposed method have the lowest quality scores implying that they are well-balanced between compactness and expressiveness.

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 [9] 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.

(a)			(b)		
S→Y	(8.00)	[0.53]	S→AABCC	(6.99)	[0.60]
| AYC	(3.00)	[0.20]	| ASC	(2.66)	[0.23]
| AABAC	(1.00)	[0.07]	| AASCC	(0.93)	[0.08]
| AACACCCC	(1.00)	[0.07]	| CS	(0.64)	[0.05]
| AAYCC	(1.00)	[0.07]	| AABAC	(0.46)	[0.04]
| CY	(1.00)	[0.07]			
Y→AABCC	(8.00)	[1.00]			

Fig. 5. Obtained grammars using method in [9](a) and proposed method(b) from data with noise probability 0.08.

It is worth noting that the rule scores in the grammar generated using our method reflect the uncertainty values of input symbols. As a result, in Fig. 5(b) the erroneous sequence $AABAC$ (the last rule) has a rule score of 0.46 in contrast to 1.00 in Fig. 5(a), as the symbol C had lower probability (higher 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 relatively higher probability value. This will result in higher likelihood when parsed on new samples with the same class.

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

B. The Towers of Hanoi

We try our method on real-world data obtained from the demonstrations of 5 human participants using a camera. We set our goal to be a successful imitation where a robot follows the correct sequence of actions demonstrated by a human partner. However, instead of simply imitating, we require that the robot should deal with noise using the knowledge obtained in prior so that it can perform the intended action sequence correctly even when the perceived symbols are partially incorrect. Furthermore, we are interested in tasks that include recursion which can be demonstrated in various lengths of action sequences, resulting in more challenging setting. We choose The Towers of Hanoi problem as it satisfies the above requirements.

1) Experiment Scenario : In training phase, a human demonstrator solves the puzzle using only 2 and 3 disks, repeating each 3 times. The robot then learns an activity grammar from each demonstrator using techniques explained in Sec. IV. 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 reproduction 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. Fig. 7 shows the end state of a reproduction using the iCub simulator (see associated video file in the proceedings).

We experiment under two types of noise conditions: low-noise (indoor lighting) and high-noise (direct sunlight) conditions. That is, a) train on low-noise condition and test on both low- and high-noise conditions, respectively, and b)

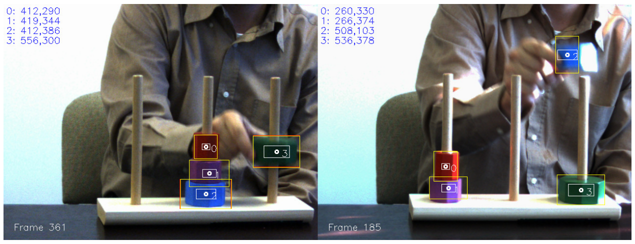

Fig. 6. A sample tracking screen while a human participant is solving the puzzle with 4 disks. Compared to the left figure (low-noise condition), the right figure (high-noise condition) shows over-exposured spots which often makes the tracker unstable. The tracker immediately resets the position if lost by searching the desired blob from the entire region of the image.

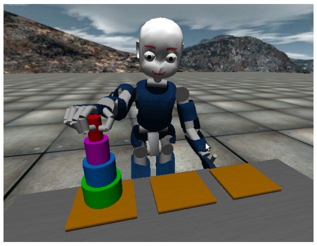

Fig. 7. The Towers of Hanoi. iCub is performing parsed actions.[15]

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

train on high-noise condition and test on both conditions. All samples of high-noise data set 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 [2], 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.

In this experiment, our system generates 5 types of action symbols during observation as detailed in Fig. 8. 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 which can avoid generating 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 the grammar building algorithm.

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 represented as $(XYZ)^n$, meaning "Perform (XYZ) recursively until the problem is solved."

We use a camera with resolution 640x480, 30 frames per second. Object trackers are implemented using standard CamShift algorithm provided in [16], 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.

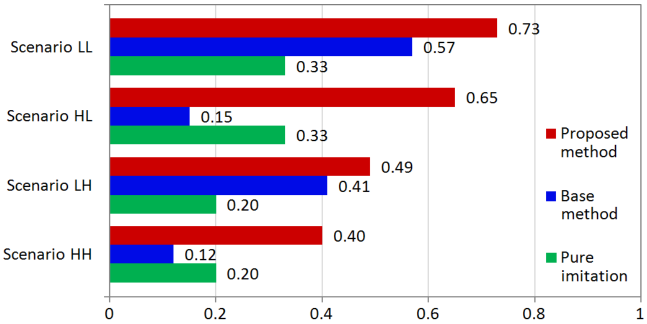

Fig. 9. Success rates using our method, base method [8] and pure imitation. Scenarios LL and LH: Train on low-noise condition and test on low- and high-noise conditions, respectively. Scenarios HL and HH: Train on high-noise condition and test on low- and high-noise conditions, respectively. The fact that a single mistake while parsing a long test sequence causes a failure makes this problem non-trivial.

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
	Imitation	25	N/A		Imitation	15	N/A
HL	Proposed	49	306.25	HH	Proposed	30	306.66
	Base	11	469.32		Base	9	469.46
	Imitation	25	N/A		Imitation	15	N/A

Fig. 10. Detailed results with average MDL scores for comparison. Each case is tested on 75 sequences. MDL score is not available for pure imitation as it does not rely on any learned model. It is worth noting that lower MDL score generally leads to higher success rate.

2) *Results and Analysis:* 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 the unforeseen, possibly more complicated activities that shares of same action components. We report the performances in 4 scenarios (LL, LH, HL, HH) in Fig. 9.

In scenarios LL and LH, models are both trained from demonstrations of 2 and 3 disks under low-noise condition, where they are tested on demonstrations of 4 disks on low-noise and high-noise conditions, respectively. Similarly, scenarios HL and HH are trained from high-noise condition and tested on both noise conditions.

We compare with the base method [8] and 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. 11 to see error statistics. We do not use the method proposed by

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

Fig. 11. Error statistics of demonstrations using 4 disks on each noise condition. Note that even in low-noise condition, there are only 5 trials observed with all correct symbols, which means that in most cases 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.

(a)		(b)	
S→LAD	[0.205669]	S→LADLBDLCD	[0.666667]
| LBD	[0.204606]	| SSLAD	[0.285714]
| LCD	[0.163233]	| SSSSS	[0.047619]
| CADSS	[0.020551]		
| SLBAS	[0.017184]		
| SSS	[0.388758]		

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 constructed in an ideal case where no noise symbols exist.

Kitani et al[9] in this experiment as all generated symbols are always related to the task.

As can be seen in Fig. 9, it is important to note that there is a noticeable difference on 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 pure imitation.

It is also worth noting that from Fig. 10, 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 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 symbols in input sequences, the underlying structures of action components are captured effectively. It is worth emphasizing that this structure enables the contextually consistent parsing of new 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 active substring searching explained in Sec. IV-A. Fig. 12(b) shows an example grammar constructed from data that contains no noise at all. Most of the experiments, however, includes noise symbols in the middle of input sequences 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.

VI. Conclusions and Future Directions

We have presented an activity grammar learning method which searches for frequently occurring sub-strings from the input stream that are likely to be reusable considering the uncertainty values of the input symbols. We have shown in 4 non-trivial real-world scenarios that our method is capable to learn reusable structures under reasonable amount of noise. We have experimentally shown that a lower MDL score generally leads to higher performance on parsing unforeseen actions.

The discovery of important component actions and recursions were critical to the performance which are supported by the results reported in Sec. V-B.2. For example, the action component LAD in Fig. 12(a), line 1 (Lift a disk, move between towers 1 and 2, then drop it.) allows contextually consistent interpretation by biasing the parser to parse in the order of $L - A - D$ even when the observed symbols are partially wrong. The results reported in Sec. V-A support our idea that handling uncertainty values of input symbols is beneficial. It is worth mentioning that we have not used any prior information at the symbolic level.

In future work, the inclusion of structural priors could be 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 the tasks.

Acknowledgments

This work was supported by the EU FP7 project EFAA (FP7-ICT-270490).

References

[1] M. Ryoo and J. Aggarwal, "Robust human-computer interaction system guiding a user by providing feedback," in *IJCAI*, 2007, pp. 2850–2855.

[2] Y. A. Ivanov and A. F. Bobick, "Recognition of visual activities and interactions by stochastic parsing," *TPAMI*, vol. 22, pp. 852–872, 2000.

[3] K. Lee and Y. Demiris, "Towards incremental learning of task-dependent action sequences using probabilistic parsing," *ICDL*, 2011.

[4] C. de la Higuera, "A bibliographical study of grammatical inference," *Pattern Recognition*, vol. 38, no. 9, pp. 1332–1348, 2005.

[5] Y. Aloimonos, *et al.*, "The language of action: a new tool for human-centric interfaces," *Human Centric Interfaces for Ambient Intell.*, 2009.

[6] C. Nevill-Manning and I. Witten, "Identifying hierarchical structure in sequences: A linear-time algorithm," *JAIS*, pp. 67–82, 1997.

[7] Z. Solan, *et al.*, "Unsupervised learning of natural languages," *PNAS*, vol. 102, no. 33, pp. 11 629–11 634, 2005.

[8] A. Stolcke and S. Omohundro, "Inducing probabilistic grammars by bayesian model merging," *Gramm. Infer. and App.*, pp. 106–118, 1994.

[9] K. Kitani, *et al.*, "Recovering the basic structure of human activities from noisy video-based symbol strings," *IJPRAI*, 2008.

[10] Y. Kuniyoshi, *et al.*, "Learning by watching: Extracting reusable task knowledge from visual observation of human performance," *T. Robotics and Automation*, vol. 10, pp. 799–822, 1994.

[11] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in Cognitive Sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[12] Y. Demiris and B. Khadhouri, "Hierarchical attentive multiple models for execution and recognition of actions," *RAS*, vol. 54, no. 5, 2006.

[13] P. Langley and S. Stromsten, "Learning context-free grammars with a simplicity bias," in *ECML*. Springer, 2000, pp. 321–338.

[14] F. Zhou, *et al.*, "Aligned cluster analysis for temporal segmentation of human motion," in *Automatic Face & Gesture Recognition*, 2008.

[15] U. Pattacini, *et al.*, "An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots," in *IROS*, 2010.

[16] G. Bradski, "The opencv library," *Doctor Dobbs Journal*, vol. 25, no. 11, pp. 120–126, 2000.