THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

TECHNICAL REPORT OF IEICE.

An MDL Approach to Learning Activity Grammars

Kris M. KITANI†, Yoichi SATO†, and Akihiro SUGIMOTO††

† Institute of Industrial Science, The University of Tokyo, 4–6–1 Komaba, Meguro, Tokyo 153–8505 JAPAN
†† National Institute of Informatics, 2–1–2 Hitotsubashi, Chiyoda, Tokyo 101–8430 JAPAN
E-mail: †{kitani,ysato}@iis.u-tokyo.ac.jp, ††sugimoto@nii.ac.jp

Abstract Stochastic Context-Free Grammars (SCFG) have been shown to be useful for vision-based human activity analysis. However, action strings from vision-based systems differ from word strings, in that a string of symbols produced by video contains noise symbols, making grammar learning very difficult. In order to learn the basic structure of human activities, it is necessary to filter out these noise symbols. In our work, we propose a new technique for identifying the best subset of non-noise terminal symbols and acquiring the best activity grammar. Our approach uses the Minimum Description Length (MDL) principle, to evaluate the trade-offs between model complexity and data fit, to quantify the difference between the results of each terminal subset. The evaluation results are then used to identify a class of candidate terminal subsets and grammars that remove the noise and enable the discovery of the basic structure of an activity. In this paper, we present the validity of our proposed method based on experiments with synthetic data.

Key words Grammatical Inference, Syntactic Analysis, Minimum Description Length Principle, Action Recognition

1. Introduction

The Context-Free Grammar (CFG) is a model that has been widely utilized for natural language processing. However in recent years, it has been shown that CFGs are also effective for modeling human activities extracted from video ([1], [7], [3], [2]). The success of CFGs in analyzing natural languages, is largely due to its ability to represent the hierarchical structure found among words in a sentence. This same hierarchical structure is also found in human activities [8] which makes an CFG a suitable model for human activity analysis. While a variety of other non-hierarchical finite-state machines (finite-state automata, hidden Markov models, n-grams, etc.) have been used for human activity analysis, they are limited in that they cannot describe hierarchical structures.

One important task involved in using a CFG for activity analysis is the task of *defining* the grammar. Previous works that used CFGs for activity analysis defined their grammars by hand-crafting their own grammars and have left the issue of grammar learning unaddressed. Ivanov [1] extracted primitive action words from a video sequence of a conductors arm using HMMs and was able to recognize the rhythmic meter using a Stochastic Context-Free Grammar (SCFG). The grammar and it's parameters however, were defined by Ivanov himself. Moore [3] used a SCFG to recognize people playing Black Jack and used the *a prior* information encoded in the grammar to deal with errors in the string of action words. Again, the grammar was defined by the author based on the basic rules of the game. In the same vein, Minnen [2] leveraged the *a prior* knowledge of a predefined grammar to infer an action even when the agent under analysis is occluded in the scene.

In contrast to the number of works that used predefined grammars, research dealing with the issue of learning has been minimal. Wang [7] used a similar experimental scenario as Ivanov and implemented HMMs to produce action words from a video segment of a conductors hand motions. The actions produced by the HMMs were then fed into a CFG learning algorithm COMPRESSIVE [4] to learn the action grammar. Due to the fact that COMPRESSIVE presupposes positive examples to generate the CFG, the system is very sensitive to errors in the training data. That is, an unstable detection would have a very adverse effect on the learning process because the noise would be included into the learned grammar. While a noise-less input stream may be a reasonable assumption when learning a grammar from a string of words, it is a naive assumption when learning an activity grammar from a symbol string that has been generated using probabilistic detectors on a noisy video source.

In summary, most of the work using CFGs for activity analysis have used grammars designed by knowledge engineers while research focused on grammar learning have only used pre-exiting algorithms, assuming activities to be a noise-less stream of symbols. Therefore, in this paper, we propose a new grammar learning method that deals with the issue of noise. Our method places an assumption of noise on the training data and tests that assumption using the minimum description length (MDL) principle. Then, using the results of the MDL evaluation, our method finds a class of the best set of terminal symbols that yields the most compact and descriptive grammars.

2. Our Approach

In this section we first discuss the characteristics of noise in 2. 1 and then we introduce the basic ideas behind our approach in 2. 2.

2. 1 Characteristics of Noise

When considering the task of extracting action symbols from video via image processing, it is reasonable to expect different types of errors in the symbol string. Due to changes in appearance, some symbols might be mistakenly *inserted* into the string, while other symbols are *deleted* (not detected) or *substituted* by a wrong symbol. In general, certain symbols that are often inserted, deleted or substituted introduce a lot of randomness into the string. It is symbols such as these that we call *noise symbols*.

In this paper, we make the assertion that a symbols is either a (1) noise symbols or a (2) non-noise symbol. We define a non-noise symbol to be a symbol that shows non-random behavior, meaning that the symbol is observed with regularity with respect to frequency and its relative position to other symbols. Noise symbols on the the other hand, display random behavior. That is, noise symbols do not appear with a constant frequency and they appear sporadically, unaffected by the presence of other symbols.

2. 2 Basic Concept

We begin the explanation of our method with a concrete example to show the basic concepts underlying our approach. Given a symbol string S, we would like to find the most *compact grammar* that yields a *detailed description* of the symbol string. At first glance, no regularity is observed in the string:

$$\texttt{S} \rightarrow \texttt{a x b y c a b x c y a b c x}.$$

Since we are assuming the presence of noise, we randomly remove all `y` symbols from the stream making the *presupposition* that it is noise[1]. This assumption allows us to shrink the string into its new form:

$$\texttt{S} \rightarrow \texttt{a x b c a b x c a b c x}.$$

It is observed that the substring `c a b` occurs twice in the string but we still have not found a *regularity* (some rule) that completely describes the symbol string. So we proceed by making another arbitrary presupposition that `x` is also a noise symbol, resulting in the string:

$$\texttt{S} \rightarrow \texttt{a b c a b c a b c}.$$

Now it is clear that the substring `a b c` is repeated 3 times in the symbol string. Thus we create a new rule `A` and encode the symbol string S with the new rule, yielding the compact description:

$$\texttt{S} \rightarrow \texttt{A A A}$$
$$\texttt{A} \rightarrow \texttt{a b c}$$

What we have observed through this example is that when `x` and `y` are assumed *a priori* to be noise, we were able to obtain a compact grammar (`A` $\rightarrow$ `a b c`) and a detailed description of the basic structure of the original symbol string `S` $\rightarrow$ `A A A`. Thus we reason that it is highly probable that `x` and `y` are in fact noise symbols.

If, as in the example, we can correctly guess which symbols are noise symbols, we will be able to find the most compact representation of the symbol string. However, this holds true only when the following criteria are met:

(1) noise symbols exist in the symbol string,
(2) non-noise symbols exist in the symbol string,
(3) noise and non-noise symbols are mutually exclusive,
(4) non-noise symbols occur with regularity.

When these conditions are met and the noise symbols are correctly identified, we will be able to obtain a compact grammar and a detailed description of the basic structure of the symbol string. Therefore, in our approach, we will identify noise symbols by evaluating the complexity of the grammar (model) and the ability of the grammar to fully describe the symbol string (data).

The remainder of this paper is organized as follows. In the next section 3., terminology and the basic concepts of description length are introduced. In section 4., the implementation details of the proposed method are given and in 5., we show the validity of our method from experimental results based on synthetic training data.

3. Preliminaries

3. 1 Defining a SCFG and the Input Data

A SCFG is defined by the quintuplet $\mathbf{G} = \{\mathbf{T}, \mathbf{N}, S, \mathbf{R}, \mathbf{\Theta}\}$, where $\mathbf{T}$, $\mathbf{N}$, $\mathbf{R}$, $\mathbf{\Theta}$ are the set of terminal symbols $\{T_1, \ldots, T_m\}$, the set of nonterminal symbols $\{N_1, \ldots, N_n\}$, the set of production rules $\{R_1, \ldots, R_p\}$, the set of parameter of the production rules $\{\theta_1, \ldots, \theta_q\}$, respectively and S

[1] Here we delete the symbol for illustrative purposes. We do not actually delete symbols in our method.

Table 1 Pseudo-code for the algorithm

```
For every combination of terminal symbols C_k,  k = 1 to 2^m
  CreateInitialGrammar(C_k, W) → G_0
  LearnGrammar(G_0) → G
  GrammarCodelength(G) → − log P(G)
  EncodedDataCodelength(W′, G) → − log P(W′|G)
  TotalCodelength → − log P(G|W′)
OutputGrammarCandidates → Ω
```

is the start symbol, where $S \in \mathbf{N}$. A production rule takes the form $A \rightarrow \alpha^*$, where the production rule probability $P(A \rightarrow \alpha^*)$, conforms to the condition $\sum_k P(A \rightarrow \alpha_k^*)) = 1$, where $A \in \mathbf{N}$, $\alpha \in (N_i \cup T_j)$ and $N_i \in \mathbf{N}$.

The input symbol string (training data) $\mathbf{W} = \{W_1, \ldots, W_u\}$ is made of u action sequences W_i extracted from a long video sequence. Furthermore, the action sequence $W_i = \{\epsilon_0, w_1, \ldots, w_v\}$, is made of v action words (terminal symbols) w_j, where each sequence is headed by a ϵ_0 that demarks the beginning of a new sequence.

3.2 Relation between MDL and Bayes Rule

In the Bayesian framework, the optimal model $\mathbf{G}$ is found by maximizing the product of the data likelihood $P(\mathbf{W}|\mathbf{G})$ and the prior probability $P(\mathbf{G})$ of the model, formally:

$$\hat{\mathbf{G}} = \arg\max_G \{P(\mathbf{W}|\mathbf{G})P(\mathbf{G})\}. \quad (1)$$

In contrast, in terms of information-theoretic concepts, a probability has another interpretation. That is, given a discrete random variable $\mathbf{X}$ and an event x with a probability distribution $P(x)$, the number of bits needed to describe that event x is given by the Shannon code:

$$DL(x) = -\log_2 P(x). \quad (2)$$

Thus by taking the minus log of the argument (1) we obtain a formulation of the minimum description length principle:

$$\hat{\mathbf{G}} = \arg\min_G \{-\log_2 P(\mathbf{W}|\mathbf{G}) - \log_2 P(\mathbf{G})\}. \quad (3)$$

In the MDL framework, the optimal model $\hat{\mathbf{G}}$ is found by minimizing the description length of the model $-\log P(\mathbf{G})$ and the description length of the data encoded by the model $-\log P(\mathbf{W}|\mathbf{G})$.

4. Proposed Method

In this section we give a detailed explanation of the flow of the algorithm as summarized in table 1.

4.1 For each Combination $\mathbf{C_k}$

Since we do not know *a priori* which symbols are noise, we evaluate every possible combination of terminal symbols, such that a combination of terminal symbols expresses a certain presupposition (noise or non-noise) about the symbols. For example, if we have two symbols `x` and `y`, there are four possible combinations (four ways to divide the symbols). The complete set of possible non-noise symbols would be $\{\emptyset\}, \{x\}, \{y\}$ and $\{x, y\}$. We formalize this below.

Given m terminal symbols, it is possible to produce 2^m different combinations. Each combination represents a presupposition about the nature of each symbol. Formally, combination $\mathbf{C}_k$ is defined by two sets, $\mathbf{T}^+ = \{T_1^+, \ldots, T_u^+\}$ and $\mathbf{T}^- = \{T_1^-, \ldots, T_v^-\}$, such that $k = (1, \ldots, 2^m)$ and $u + v = m$. The set $\mathbf{T}^+$ is the presupposed set of all non-noise symbols, where each element T_i^+ is called an informative symbol. Similarly, the set $\mathbf{T}^+$ is the set of all presupposed noise symbols, where each element T_i^- is called a non-informative symbol. The two sets have the property such that $\mathbf{T} = \mathbf{T}^- \cup \mathbf{T}^+$ and $\mathbf{T}^- \cap \mathbf{T}^+ = \emptyset$.

4.2 Creating the Initial Grammar $\mathbf{G_0}$

In the previous section, we set up presuppositions for each terminal symbol and split up the terminal symbols into two subsets. Before we start the MDL evaluation process, we must create an initial grammar that reflects our presuppositions about the terminal symbols.

First we create a set of preterminal production rules $N_i \rightarrow T_i^+$ for each element in $\mathbf{T}^+$, where N_i is a newly created nonterminal. These preterminal rules effectively preserve the unique identity of the symbol in the input string.

Next, we create a set of generic preterminal production rules for non-informative symbols in the form $* \rightarrow T_i^-$, where the nonterminal $*$ is a generic nonterminal representing all noise symbols. The generic absorption rule $* \rightarrow * *$ is also created, which has the effect of absorbing a series of adjacent noise symbols.

Finally, we create a rule that contains the whole input symbol string $\mathbf{W}$ headed by S, since the next grammar learning step requires that the whole input string be the first rule in grammar. The input symbol string $\mathbf{W}$ is a series of segmented activity sequences, where each sequence W_i is headed by a start marker ϵ_i. In our work, segmentation is done manually for simplicity. However, there are various ways to segment activities in video, such as using a threshold on frame to frame differences.

Before we add the input symbol string to the grammar, we would first like to partially encode the input string, to reflect the presuppositions that we have made about the symbols. We do this by replacing each terminal symbol w_j in each sequence $W_i \in \mathbf{W}$ with the appropriate nonterminal symbol, using the preterminal productions rules created earlier. The partially encoded input string $\mathbf{W}'$ is then used to create the new rule $S \rightarrow \mathbf{W}'$. After all of the new rules have been inserted into the grammar, we obtain the set of initial rules $\mathbf{R}_0$:

$$\mathbf{R}_0 = \left\{ \begin{array}{ll} S \rightarrow \mathbf{W}' & * \rightarrow ** \\ N_1 \rightarrow T_1^+ & * \rightarrow T_1^- \\ \cdots & \cdots \\ N_u \rightarrow T_u^+ & * \rightarrow T_v^- \end{array} \right\}.$$

4.3 Learning the Grammar

In order to test the presuppositions regarding the symbols, we learn the grammar based on the presuppositions and use the learned grammar as a measure of correctness. We use Nevill-Manning's COMPRESSIVE algorithm [4] to learn the grammar, using the initial grammar $\mathbf{G}_0$.

COMPRESSIVE uses a formula that quantifies the change in description length ΔDL to find the best *N-gram* in the grammar that minimizes the overall size of the grammar. For a *N-gram* ν with length m and occurrence n, the compression function is given as:

$$\hat{\nu} = \arg\max_{\nu} \{m \cdot n - (m+1) - n\}. \tag{4}$$

In words, the change in description length is equivalent to the decrease caused by the removal of ν (n occurrences of length m), minus the increase of inserting a new rule $m+1$, minus the increase of inserting of the new nonterminal symbol n times.

Once the best ν has been found and replaced by the new nonterminal, the algorithm reprocesses the grammar until there are no more *N-grams* can be found that decrease the size of the grammar. During the iterative process, the occurrence counts for the best *N-grams* are stored and are used later to calculate the rule probabilities.

Upon completion of COMPRESSIVE, the grammar is post-processed and probabilities are calculated for each rule. First, the S rule containing the compressed input string $\mathbf{W}''$ is separated back into the segments W_i'' using the markers ϵ_i and inserted back into the grammar as a new S rule ($S \rightarrow W_i''$). Multiple occurrences of the same segment are inserted only once while the counts are continually updated.

After post-processing, the production rule probabilities are calculated with the follow equation:

$$P(A \rightarrow \lambda_i^*) = \frac{c(A \rightarrow \lambda_i^*)}{\sum_j c(A \rightarrow \lambda_j^*)}, \tag{5}$$

such that $A \in \mathbf{N}$ and $\lambda = (N_p \cup T_q)$.

4.4 Description Length of the Grammar

Once the grammar has been learned, we quantify it by calculating the description length of the grammar. Later we will see how this description length is used to evaluate the overall correctness of the presuppositions on the symbols.

We used the same method as Stolcke [6] to calculate the description length of the grammar. Since the probability of the grammar is defined to be the joint probability of the parameters and structure of the grammar $P(\mathbf{G}_S, \mathbf{\Theta_G}) = P(\Theta_\mathbf{G}|\mathbf{G}_S)P(\mathbf{G}_S)$, we divide the computation of the description length into two parts.

$$DL(\mathbf{G}) = DL(\Theta_\mathbf{G}|\mathbf{G}_S) + DL(\mathbf{G}_S) \tag{6}$$

To calculate the description length of the grammar, we use a probabilistic interpretation for the parameters and a code-length interpretation for the structure.

Grammar Parameters Θ_G

The grammar parameter probability is calculated as the product of Dirichlet distributions $P(\Theta_G|\mathbf{G}_S) = \prod_{j=1}^{n} P(\theta_{N_j}|\mathbf{G}_S)$, such that each Dirichlet distribution represents an equally distributed probability across all potential productions of a given nonterminal. For a nonterminal N,

$$P(\theta_N|\mathbf{G}_S) = \frac{1}{B(\alpha_1, \ldots, \alpha_t)} \prod_{i=1}^{t} \theta_i^{\alpha_i - 1}, \tag{7}$$

where the parameters of a given nonterminal is represented by the multinomial distribution $\theta_N = (\theta_1, \ldots, \theta_t)$. Each rule has a equality distributed probability θ_i and the prior weights α_i are also equally distributed, conforming to the conditions $\sum_{i=1}^{n} \alpha_i = 1$ and $\alpha_i < 1$. The probability is converted to the description length $DL(\Theta_\mathbf{G}|\mathbf{G}_S)$ using equation 2.

Grammar Structure G_S

In contrast to the parameters, the structure is directly calculated as a description length consisting of two parts: (1) the length of the production rule *code*(k) and (2) the number of symbols in the production rule *code*(s). *code*(k) is computed from equation 8 assuming the length of the production rule is assumed to be drawn from a Poisson distribution(we use $\eta = 3$) shifted by one since the smallest possible rule is of length two.

$$-\log p(k-1;\eta) = -\log \frac{e^{-\eta}\eta^{k-1}}{(k-1)!}. \tag{8}$$

Assuming all symbols have the same occurrence probability, we need at least $-\log \frac{1}{|\Sigma|}$ bits per symbol, such that $\Sigma = \mathbf{N} \cup \mathbf{T}$. Therefore, *code*($s$) of a rule with k symbols requires $k \log |\Sigma|$ bits to describe. The total description length of the structure is:

$$DL(G_S) = \sum_{R \in \mathbf{R}} \left(-\log p(k-1;\eta) + k \log |\Sigma| \right). \tag{9}$$

4.5 Input Data and Total Description Length

Using the learned grammar, we now calculate the description length of the input data encoded by the grammar $DL(\mathbf{W}|\mathbf{G})$. $DL(\mathbf{W}|\mathbf{G})$ is another aspect of our test for correctness, that is, if our presuppositions are correct, the description length of the input data encoded by the grammar will be short.

We use a chart of inside probabilities to calculate the likelihood using the initialization equation (10, 11) and the recursive equation (12), where N is a nonterminal, i is the start

index, j is the length and k is the abstraction level of E. The summation term is the sum of inside probabilities for every permutation of $\{j_1, \ldots, j_m\}$ that sums to j.

$$\beta(T, i, 1, 1) = 1.0 \tag{10}$$

$$\beta(N, i, 1, 2) = P(N \rightarrow T) \cdot \beta(T, i, 1, 1) \tag{11}$$

$$\beta(N, i, j, k) = P(N \rightarrow N_1 \cdots N_m) \cdot \sum_{m P_m} \{\beta(N_1, i_1, j_1, k_1) + \cdots + \beta(N_m, i_m, j_m, k_m)\} \tag{12}$$

The likelihood for one sequence W_i is calculated from equation (13) the sum of all S inside probabilities that start at index one and the total likelihood is given by equation (14). The likelihood is converted to a description length using equation 2.

$$P(W_i|\mathbf{G}) = \sum_k \beta(S, 1, j_{max}, k), \tag{13}$$

$$P(\mathbf{W}|\mathbf{G}) = \prod_{i=1}^{n} P(W_i|\mathbf{G}). \tag{14}$$

Finally, we obtain the total description length of the grammar and the data using the MDL equation (3). This description length can now be used as a score for the presuppositions that were first imposed on the symbols. Thus, we now have a framework to rank the *correctness* of the combination $\mathbf{C}_k$.

4.6 Grammar Candidates Ω

Up to this point, the description length has been computed mechanically, yielding a method for scoring a combination of terminal symbols. However, when we consider the perspective of the user, a ranked list of grammars is not the best way to present the candidate grammars. Often times, the user might wish to specify the number of terminals to use, with a willingness to gaining expressiveness at the cost of an increase in description length. Therefore, in this system, we make the user-centered decision to output a set of candidate grammars such that each grammar is the top scoring grammar, for a given number of utilized terminal symbols (informative symbols). The user is then given the final authority to choose the best grammar from those candidates.

5. Experimental Results

In this section, we show the validity of our approach by learning a set of candidate grammars from synthetic training data set.

5.1 Simple Grammar with Noise

The first action grammar is a simple grammar (Table 2) that outputs one non-hierarchical sentence that describes the use of a printer. The basic pattern is `door`, `computer`, `printer` and represents the activity of a person entering from the door, sitting down at the computer and then going to the printer. We assume two unstable noise sources: (1) `desk` and (2) `shelf`, which are hypothetically produced by the image processing system when a person stands nearby either object. Sixty artificial strings were produced using the grammar.

Table 2 Predefined Grammar (S2)

S	→	DOOR	COMPUTER	PRINTER	(1.0)
DOOR	→	door	INSERT		(0.5)
DOOR	→	door			(0.5)
COMPUTER	→	computer	INSERT		(0.5)
COMPUTER	→	computer			(0.5)
PRINTER	→	printer	INSERT		(0.5)
PRINTER	→	printer			(0.5)
INSERT	→	**desk**			(0.5)
INSERT	→	**shelf**			(0.5)

Table 3 Learned Grammar for 3 Symbols (S2)

S	→	J			(0.100)	G	→	J	∗		(1.000)
S	→	L			(0.083)	H	→	B	∗	D	(1.000)
S	→	E			(0.183)	I	→	B	A	C	(1.000)
S	→	H			(0.067)	J	→	B	D		(1.000)
S	→	F			(0.167)	K	→	H	∗		(1.000)
S	→	K			(0.100)	L	→	E	∗		(1.000)
S	→	G			(0.233)	A	→	computer			(1.000)
S	→	I			(0.067)	B	→	door			(1.000)
D	→	A	∗	C	(1.000)	C	→	printer			(1.000)
E	→	B	∗	A C	(1.000)	∗	→	desk			(0.516)
F	→	I	∗		(1.000)	∗	→	shelf			(0.484)

Table 4 Candidate Combinations (S2)

SYM	Informative Symbols $\mathbf{T}^+$	$DL(G)$	$DL(W\|G)$	$DL(G\|W)$
3	computer door printer	328.62	202.137	530.756
2	door printer	117.85	400.438	518.284
1	computer	89.69	605.681	695.3758

A complete list of results for each configuration is given in Table 5[2]. The top candidates selected from this list are given in Table 4. It is observed that the true set of non-noise terminal symbols has been correctly identified when three symbols are utilized as informative symbols. The candidates for two symbols and one symbol are also given. The learned grammar when using three symbols is given in Table 2. Although the learned grammar shows a false hierarchical structure, it covers the equivalent string space as the predefined grammar in Table 2.

5.2 Three Pattern Grammar with Noise

This next grammar (Table 6) is taken from the `short meal` used in [5]. The set of non-noise terminal symbols is given as $\mathbf{T}^+ = \{$ `chair, cupboard, door, fridge` $\}$ and the set of noise terminals is given as $\mathbf{T}^- = \{$ `counter, table` $\}$.

This grammar can create 3 types of strings:

(1) `door-cupboard-fridge-chair-fridge-door`,

(2) `door-cupboard-fridge-chair-door`,

(3) `door-cupboard-fridge-door`.

Again, the proposed algorithm is executed on 60 string created by the original grammar. The results for the top candidates are given in Table 8. The true set of terminals when four symbols are used has been identified as one of the top candidate. The results for 3 symbols and below are also given.

[2] Only combinations with two or more noise symbols are shown.

Table 5 Results for All Combinations (S2)

SYM	Informative	DL(G)	DL($W|G$)	DL($G|W$)
2	door, printer	117.85	400.44	518.28
3	computer, door, printer	328.62	202.14	530.76
2	computer, door	126.81	407.14	533.95
2	computer, printer	187.57	401.41	588.98
1	computer	89.69	605.68	695.38
1	door	85.58	610.59	696.17
3	desk, door, printer	388.81	317.44	706.25
3	computer, desk, door	399.04	323.04	722.08
3	door, printer, shelf	403.31	325.53	728.84
3	computer, door, shelf	418.21	331.21	749.43
2	computer, shelf	274.87	498.50	773.37
2	computer, desk	285.39	489.49	774.88
1	printer	113.80	665.42	779.22
2	door, shelf	241.25	544.14	785.39
2	desk, door	251.33	538.11	789.44
3	computer, printer, shelf	560.90	329.26	890.15
3	computer, desk, printer	576.80	313.78	890.58
2	desk, printer	362.22	563.55	925.77
1		70.82	855.73	926.54
1	desk	173.81	761.72	935.53
2	printer, shelf	362.22	577.68	939.90
1	shelf	173.81	779.90	953.71
2	desk, shelf	459.24	642.53	1101.77
3	computer, desk, shelf	791.53	316.38	1107.90
3	desk, printer, shelf	696.86	431.66	1128.52
3	desk, door, shelf	718.64	425.67	1144.31

Table 6 `Short Meal` Grammar (C2)

S	→	ENTER	ACTION	EXIT	(0.5)
S	→	ENTER	EXIT		(0.5)
ENTER	→	DOOR	CUPBOARD	FRIDGE	(1.0)
ACTION	→	CHAIR	FRIDGE		(0.5)
ACTION	→	CHAIR			(0.5)
EXIT	→	DOOR			(1.0)
DOOR	→	door	INSERT		(0.5)
DOOR	→	door			(0.5)
CUPBOARD	→	cupboard	INSERT		(0.5)
CUPBOARD	→	cupboard			(0.5)
FRIDGE	→	fridge	INSERT		(0.5)
FRIDGE	→	fridge			(0.5)
CHAIR	→	chair	INSERT		(0.5)
CHAIR	→	chair			(0.5)
INSERT	→	**table**			(0.5)
INSERT	→	**counter**			(0.5)

Table 7 Learned Grammar for 4 Symbols (C2)

S	→	I	C		(0.02)	**S**	→	I	O			(0.02)
S	→	I	M		(0.02)	**S**	→	V	C			(0.02)
S	→	N	Q		(0.02)	**S**	→	L	M			(0.02)
S	→	X			(0.05)	E	→	G	B			(1.00)
S	→	L	P		(0.02)	F	→	E	∗	D		(1.00)
S	→	F	G		(0.03)	G	→	C	∗			(1.00)
S	→	N	M		(0.02)	H	→	C	B	D		(1.00)
S	→	I	Q		(0.03)	I	→	L	∗			(1.00)
S	→	N	G		(0.02)	J	→	C	B	∗	D	(1.00)
S	→	V	P		(0.02)	K	→	A	∗			(1.00)
S	→	F	O		(0.03)	L	→	E	D			(1.00)
S	→	R			(0.08)	M	→	A	G			(1.00)
S	→	T			(0.07)	N	→	F	∗			(1.00)
S	→	L	C		(0.03)	O	→	K	G			(1.00)
S	→	U			(0.07)	P	→	A	C			(1.00)
S	→	H	O		(0.03)	Q	→	K	C			(1.00)
S	→	S			(0.07)	R	→	I	G			(1.00)
S	→	N	O		(0.02)	S	→	F	C			(1.00)
S	→	J	C		(0.03)	T	→	H	C			(1.00)
S	→	H	M		(0.02)	U	→	H	G			(1.00)
S	→	L	Q		(0.03)	V	→	J	∗			(1.00)
S	→	H	∗	P	(0.02)	W	→	F	M			(1.00)
S	→	H	∗	G	(0.02)	X	→	L	G			(1.00)
S	→	F	P		(0.03)	Y	→	N	C			(1.00)
S	→	V	O		(0.02)	A	→	chair				(1.00)
S	→	Y			(0.05)	∗	→	counter				(0.49)
S	→	I	P		(0.02)	B	→	cupboard				(1.00)
S	→	J	G		(0.02)	C	→	door				(1.00)
S	→	V	Q		(0.02)	D	→	fridge				(1.00)
S	→	W			(0.05)	∗	→	table				(0.51)

In Table 7 the learned grammar for 4 symbols is given and can be shown to be equivalent to the original grammar.

Table 8 Top Candidates (C2)

SYM	Informative Symbols $\mathbf{T}^+$	DL(G)	DL($W|G$)	DL($G|W$)
4	chair cupboard door fridge	1182.9	397.5	1580.4
3	chair cupboard fridge	326.1	756.8	1082.9
2	cupboard door	216.9	700.6	917.4
1	door	133.0	930.2	1063.2

6. Conclusion

When one takes on the task of grammatical inference from a string of symbols produced from the results of a vision-based image processing system, the difficulty of the task is increased by the potential presence of noise symbols. Noise is introduced into the string by errors in the image processing system as well as non-regular actions symbols in the input string. In this paper, we addressed the issue of noise and presented a method for identifying the best set of non-noise terminals. Our approach was based on a type of hypothesis testing using the MDL principle to test the overall correctness of the hypothesis. We presented the details of our approach along with experimental results to show the validity of our work. We were able to show that non-noise terminals were identified correctly and resulted in learning a grammar equivalent to the original grammar used to produce the artificial input string. As for future works, we plan to test the robustness and practicality of our method by using real data.

References

[1] Y. A. Ivanov and A. F. Bobick. Recognition of Visual Activities and Interactions by Stochastic Parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):852–872, 2000.

[2] D. Minnen, I. A. Essa, and T. Starner. Expectation Grammars: Leveraging High-Level Expectations for Activity Recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages II: 626–632, 2003.

[3] D. J. Moore and I. A. Essa. Recognizing Multitasked Activities from Video Using Stochastic Context-Free Grammar. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 770–776. American Association for Artificial Intelligence, 2002.

[4] C. G. Nevil-Manning and I. H. Witten. Online and Offline Heuristics for Inferring Hierarchies of Repetitions in Sequences. In *Proceedings of IEEE*, number 11 in 88, pages 1745–1755, 2000.

[5] N. T. Nguyen, D. Q. Phung, S. Venkatesh, and H. H. Bui. Learning and Detecting Activities from Movement Trajectories Using the Hierarchical Hidden Markov Models. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages II: 955–960, 2005.

[6] A. Stolcke. *Bayesian Learning of Probabilistic Language Models*. PhD thesis, University of California at Berkeley, 1994.

[7] T.-S. Wang, H.-Y. Shum, Y.-Q. Xu, and N.-N. Zheng. Unsupervised Analysis of Human Gestures. In *Proceedings of the IEEE Pacific Rim Conference on Multimedia*, volume 2195, pages 174–181, 2001.

[8] J. M. Zacks and B. Tversky. Event Structure in Perception and Conception. *Psychological Bulletin*, 127:3–21, 2001.