International Journal of Pattern Recognition
and Artificial Intelligence
Vol. 22, No. 8 (2008) 1621–1646

World Scientific
www.worldscientific.com

RECOVERING THE BASIC STRUCTURE OF HUMAN ACTIVITIES FROM NOISY VIDEO-BASED SYMBOL STRINGS

KRIS M. KITANI* and YOICHI SATO†

Institute of Industrial Science
The University of Tokyo, Tokyo 153-8505, Japan
**kitani@iis.u-tokyo.ac.jp*
†*ysato@iis.u-tokyo.ac.jp*

AKIHIRO SUGIMOTO

National Institute of Informatics
Tokyo 101-8430, Japan
sugimoto@nii.ac.jp

In recent years stochastic context-free grammars have been shown to be effective in modeling human activities because of the hierarchical structures they represent. However, most of the research in this area has yet to address the issue of learning the activity grammars from a noisy input source, namely, video. In this paper, we present a framework for identifying noise and recovering the basic activity grammar from a noisy symbol string produced by video. We identify the noise symbols by finding the set of non-noise symbols that optimally compresses the training data, where the optimality of compression is measured using an MDL criterion. We show the robustness of our system to noise and its effectiveness in learning the basic structure of human activity, through experiments with artificial data and a real video sequence from a local convenience store.

Keywords: Activity learning; stochastic context-free grammars; grammatical inference; minimum description length principle; model selection; noise.

1. Introduction

The Stochastic Context-Free Grammar (SCFG) is a model that has been widely utilized for natural language processing and in recent years, has also been shown to be effective in modeling human activities extracted from video.[4, 6, 7, 10, 13, 16] The success of SCFGs in analyzing natural languages is largely due to its ability to represent the *hierarchical structure* found among words in a sentence. According to perceptual psychology,[17] this hierarchical structure is also characteristic of the primitive actions of a human activity[a] and like sentences, activities are perceived

[a]We use the term *activity* based on terminology introduced by Collins[2] instead of the term *event* to refer to the high-level description of a temporal sequence of primitive actions.

to have partonomic structure (a discrete temporal sequence of primitive actions). This similarity between strings of words and a series of actions gives us the rational basis for the use of an SCFG for activity analysis. Other nonhierarchical sequential state-based models (finite-state automata, hidden Markov models, n-grams, etc.) have also been successfully applied to human activity recognition but are limited by the fact that they do not explicitly describe the hierarchical structure of human activities.

One important task involved in using an SCFG for activity analysis is the task of *learning* the grammar. Most of the previous works however, have manually designed their own grammars and have avoided the issue of grammar learning. Ivanov[4] extracted primitive action words from a video sequence of a conductors arm using HMMs and was able to recognize the rhythmic meter using an SCFG. The grammar and its probabilities however, were defined by Ivanov. Moore[7] used an SCFG to recognize people playing Black Jack and used *a priori* 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. Similarly, Minnen[6] leveraged *a priori* knowledge of a predefined grammar to infer an action when the agent under analysis is occluded in the scene.

In contrast to works that used manually defined grammars, research dealing with the issue of automated learning has been minimal and assumes a pure data set for learning. Wang[16] used an experimental scenario similar to Ivanov and implemented HMMs to produce primitive action symbols from a video segment of a conductor's hand motions. The primitive actions produced by the HMMs were then fed into a pre-existing CFG learning algorithm COMPRESSIVE[9] to learn the activity grammar. Due to the fact that COMPRESSIVE requires positive examples to generate the CFG, it can be shown that their system is very sensitive to noise in the input symbol string. That is, an unstable detector or an unrelated action would have an adverse affect on the learning process because this 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 produced by stochastic detectors from a highly variable action sequence created by human actors.

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

2. Conceptual Example

We begin the explanation of our method with a conceptual 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 would like to remove different combinations of symbols to see if we can discover the underlying pattern. Here, for the sake of example we conveniently make the hypothesis that `y` is noise and remove all `y` symbols from the string.[b] 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 hypothesis 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 three times in the symbol string and so, 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` were correctly assumed *a priori* to be noise, we were able to obtain a compact grammar (`A` $\rightarrow$ `a b c`) and a deterministic description of the basic structure of the original symbol string as `S` $\rightarrow$ `A A A`. The technical formulation of the concepts and methodology introduced here are given in the following section.

3. Preliminaries

3.1. *Our focus*

It is necessary to first understand the focus of our proposed approach before we proceed to explain its details. First, our primary interest is high-level grammatical inference of human activities and not the methods for low-level primitive action extraction. Our method assumes a reliable low-level processing system that returns a string of primitive action symbols. Second, we deal with a strictly sequential string of primitive action symbols as our input. We recognize that while most activities

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

are sequential streams of primitive actions, intra-action relationships can also have other modes,[1] such as overlapping and concurrence. As such, other methods such as propagation networks,[14] Petri networks,[3] deleted interpolation[5] and CFGs[13] have been proposed to address different temporal modes between primitive actions and activities for the *recognition* task. In contrast, in regards to the *learning* task, we believe that discovering the basic sequential structure between key actions is the first important step in establishing a strong *context* to discover other types of temporal modes. Here we leave the issue of learning nonsequential temporal relationships for future work and focus primarily on discovering basic sequential patterns. Therefore, given our focus, we limit our discussion to the discovery of the grammar of a strictly sequential string of key action symbols.

3.2. *Definition of noise*

When considering the task of learning an activity from a string of action symbols, it is reasonable to expect different types of noise that might hide the basic structure of the activity that is to be learned. The first type of noise is inherent to human activities which we call *inherent noise*. Inherent noise is caused by superfluous actions that do not play an important role in defining the activity to be learned. These secondary action symbols (noise symbols) tend to appear with irregular frequency and order, and fill in the gaps between the important action symbols. The second type of noise is *system noise* caused by the instability of the image processing system. System noise can be attributed to changes in appearance that cause the image processing system to insert, substitute or delete (miss) random symbols from the symbol string. Symbols that are often inserted, substituted or deleted should not be used for learning because they introduce much randomness to the symbol string.

Since it is a very challenging task to address all the different modes of noise, we make several key assumptions to narrow our focus upon a more manageable subproblem, namely, inherent insertion noise in the training data. First, we make the assertion that a symbol is either a noise symbol or a non-noise symbol (a symbol cannot be noise and non-noise at the same time). Second, we define a non-noise symbol to be a primary action symbol that defines the target activity. As for its properties, it shows regularity in its appearance and is observed with constant frequency and ordering. Noise symbols on the other hand are secondary action symbols that display random behavior with respect to frequency and ordering. Our assumptions are summarized as follows:

(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.

While our primary assumption is that of inherent insertion noise, we also show in Sec. 5.1.2 how our method performs when these assumptions are violated using strings corrupted by inherent insertion noise and system noise.

3.3. *Context-free grammar*

As mentioned before, a context-free grammar (CFG) is used here to model human activity because of its ability to explicitly and compactly describe hierarchal structure. A CFG is defined by the 4-tuple $\mathbf{G} = \{\mathbf{T}, \mathbf{N}, S, \mathbf{R}\}$, where $\mathbf{T}$ is a finite set of terminal symbols, $\mathbf{N}$ is a finite set of nonterminal symbols, S is the start symbol (a special nonterminal symbol) and $\mathbf{R}$ is the set of production rules. The production rules take the form $A \rightarrow \lambda^*$, which states that nonterminal symbol A produces the string λ^* of one or more symbols. When a probability $P(A \rightarrow \lambda^*)$ that satisfies the condition $\sum_i P(A \rightarrow \lambda_i^*) = 1$, is associated to each rule, the grammar becomes a stochastic content-free grammar (SCFG).

When a SCFG is used to model activity, each terminal symbol represents a primitive action and each nonterminal symbol represents an abstraction of a substring of terminal symbols. The start symbol S represents a single activity, a complete symbol string produced by the grammar.

4. Proposed Method

In this section, the key concepts introduced through the conceptual example in Sec. 2 are formalized and it is shown how the MDL principle can be used to identify the correct noise symbols.

4.1. *Setting up the presuppositions*

To learn a grammar from the training data, it is required to first remove any noise that might be contained in the training data. Formally, given the training data $\mathbf{W} = \{W_1, \ldots, W_l\}$, a concatenation of l activity sequences W_i, where each activity sequence $W_i = \{w_1, \ldots, w_p\}$ is a string of primitive action symbols $w_j \in \mathbf{T}$, it is our goal to identify the symbols that are not useful (noise) for learning the grammar. However, since we do not know *a priori* which symbols are noise, we propose to set up various presuppositions (noise or not noise) against each unique primitive symbol and evaluate that hypothesis using an MDL criterion. Here we explain how a single presupposition or hypothesis is set up.

A single hypothesis divides the set of primitive actions (terminal symbols) into two sets: the set of noise symbols $\mathbf{w}^f = \{w_1^f, \ldots, w_v^f\}$ and the set of non-noise symbols $\mathbf{w}^t = \{w_1^t, \ldots, w_u^t\}$. Next, an initial grammar is constructed to reflect the hypothesis. The initial grammar given in its general form is the set of

production rules

$$\mathbf{R}_0 = \left\{ \begin{array}{c} S \rightarrow \mathbf{W}' \\ N_1 \rightarrow w_1^t \\ \vdots \\ N_u \rightarrow w_u^t \\ \eta \rightarrow \eta\eta \\ \eta \rightarrow w_1^f \\ \vdots \\ \eta \rightarrow w_v^f \end{array} \right\}. \tag{1}$$

The first rule of the form $S \rightarrow \mathbf{W}'$ is the start production rule. S is a nonterminal symbol that represents all possible symbol strings produced by the grammar and in the initial stage $\mathbf{W}'$ is the concatenated training data encoded by the other production rules of the initial grammar. To attain the encoded input symbol string $\mathbf{W}'$, we begin with the plain input symbol string $\mathbf{W}$ and encode the plain input string to reflect the presuppositions made about each terminal symbol. This is done by replacing each terminal symbol w_i with the appropriate nonterminal symbol using the preterminal production rules, which are defined next.

The set of production rules of the form $N_i \rightarrow w_i^t$ is created for each presupposed non-noise symbol, where w_i^t is a non-noise terminal symbol and N_i is a newly created nonterminal. These preterminal rules effectively preserve the unique identity of the symbol in the training data.

The set of generic preterminal production rules of the form $\eta \rightarrow w_j^f$, where w_j^f is a noise terminal symbol and the nonterminal η is a generic nonterminal representing all noise symbols. The generic absorption rule $\eta \rightarrow \eta\,\eta$ is also created, which encodes a series of adjacent noise symbols. An example of setting up a hypothesis in the form of an initial grammar is given in Fig. 1.

4.2. *Learning the hypothesis grammar*

Now that the presuppositions on the primitive action symbols have been encoded into the initial grammar, we proceed to learn the hypothesis grammar. This initial grammar is called the hypothesis grammar because it reflects a hypothesis (presupposition) about which symbols are noise and which symbols are not noise. In later sections, it is shown how each hypothesis is tested by measuring the expressive power of each hypothesis grammar.

The heuristic CFG learning algorithm COMPRESSIVE is implemented to learn the grammar. When the original (hidden) grammar conforms to certain constraints and there is sufficient training data, the algorithm is able to learn a grammar that is very similar to the original grammar. Four assumptions made regarding

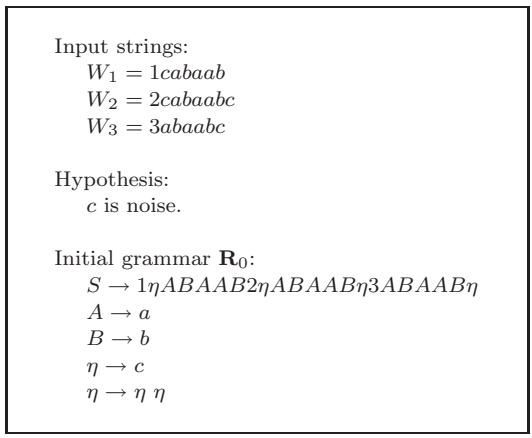

Fig. 1. Setting up a presupposition.

the original grammar are: (1) there are no cyclic (recursive) rules in the grammar, (2) there are no alternative expansions for nonterminals (only one expansion for a given nonterminal), (3) there are no abstractions (the number of symbols on the left-hand side of a rule is never 1) and (4) the grammar is optimal with respect to its description length.

When the original grammar does not conform to these assumptions, the grammar learned by the algorithm tends to be more complex (have more production rules) than the original grammar. However, since it is later shown that we are primarily concerned with identifying the hypothesis that minimizes the overall description length, we are more concerned about the relative difference in complexity between hypothesis grammars rather than their absolute similarity to the original grammar.

COMPRESSIVE uses a function that quantifies the change in description length ΔDL to find the best *n-gram* in the grammar that minimizes (compresses) the overall size of the grammar. For a *n-gram* ν with length n_ν and occurrence m_ν, the function is given as

$$\Delta DL = n_\nu \cdot m_\nu - (n_\nu + 1) - m_\nu. \tag{2}$$

In words, the change in description length is equivalent to the decrease caused by the removal of ν (m occurrences of length n), minus the increase of inserting a new rule $n+1$, minus the increase of inserting of the new nonterminal symbol m times. An example is given in Fig. 2.

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

Input symbol string:

$S \rightarrow a\ \ b\ \ c\ \ d\ \ a\ \ b\ \ c\ \ d\ \ b\ \ c\ \ d\ \ a\ \ b\ \ a\ \ b$

Pattern ν	Occurrence Frequency n_ν	Length m_ν	Compression Factor ΔDL
$b\ \ c\ \ d$	**3**	**3**	**2**
$a\ \ b\ \ c\ \ d$	2	4	1
$a\ \ b$	4	2	1
$c\ \ d$	3	2	0

(1) Replace N-gram with maximum compression as new rule:

$A \rightarrow b\ \ c\ \ d$

(2) Encode input:

$S \rightarrow a\ \ A\ \ a\ \ A\ \ A\ \ a\ \ b\ \ a\ \ b$

(3) Repeat steps (1) and (2)

Fig. 2. An example of COMPRESSIVE.

Upon completion of COMPRESSIVE, the grammar is post-processed. Recall that the original segmented input symbol string $\mathbf{W}$ was encoded by the presuppositions to acquire $\mathbf{W}'$. Now after the completion of the COMPRESSIVE algorithm, the input string has been compressed to its new form $\mathbf{W}''$. In the post-processing step, we revert $\mathbf{W}''$ back to its original l activity sequences and group sequences that have the same structure. To do this, we first remove the S rule, $S \rightarrow W_1'' \cdots W_l''$, from the grammar. Next, we separate each sequence and place a new S rule for each unique sequence $S \rightarrow W_1'', \ldots, S \rightarrow W_h''$ back into the grammar. Since unique sequences are only inserted once into the grammar $h \leq l$, the probability for each production rule is calculated by the following equation:

$$P(N \rightarrow \lambda_i^*) = \frac{c(N \rightarrow \lambda_i^*)}{\sum_j c(N \rightarrow \lambda_j^*)}, \tag{3}$$

such that N is a nonterminal, λ^* is the right-hand side of the rule and $c(\cdot)$ is a count function. Rules with zero probability are removed from the grammar.

This completes the step for learning the hypothesis grammar based on the initial presuppositions. The next section explains the framework used to evaluate the quality of the hypothesis grammar.

4.3. *Testing using the MDL principle*

We wish to find a presupposition on the primitive action symbols that gives us a *compact grammar* and a *detailed description* of the input symbol string. Reworded in the framework of MDL, we are looking for a selection of non-noise symbols that will give us a grammar $\mathbf{G}$ that minimizes the sum of the description length of the

grammar $DL(\mathbf{G})$ and the description length of the data encoded by the grammar $DL(\mathbf{W}|\mathbf{G})$ (data log-likelihood).

$$\hat{\mathbf{G}} = \arg\min_{G}\{DL(\mathbf{G}) + DL(\mathbf{W}|\mathbf{G})\} \tag{4}$$

$$= \arg\min_{G}\{-\log P(\mathbf{G}) - \log P(\mathbf{W}|\mathbf{G})\}. \tag{5}$$

In this section, we use the encoding technique proposed by Stolcke[15] to find the description length of the grammar and we use inside (beta) probabilities introduced by Pynadath[11] to calculate the description length of the data likelihood.

4.3.1. *Description length of the grammar*

The first term of the MDL equation is the description length of the grammar $DL(\mathbf{G})$. $DL(\mathbf{G})$ is a measure of the compactness of the grammar and is an indicator of the *regularity* found in the training data.

Since the probability of the grammar can be interpreted as the joint probability of the *parameters* θ_G and *structure* G_S of the grammar,

$$P(\mathbf{G}) = P(G_S, \theta_G) = P(\theta_G|G_S)P(G_S), \tag{6}$$

the description length of the grammar can be acquired by summing up the description length of the grammar parameters $DL(\theta_G|G_S)$ and the description length of the grammar structure $DL(G_S)$. We solve for $DL(\theta_G|G_S)$ using the parameter probability $P(\theta_G|G_S)$ and find $DL(G_S)$ directly from the grammar.

First, the prior on the grammar parameters $P(\theta_G|G_S)$ is calculated as the product of Dirichlet distributions [Eq. (7)], such that each Dirichlet distribution represents an uniformly distributed probability across all q possible productions of a nonterminal symbol N.

$$P_N(\theta_G|G_S) = \frac{1}{B(\alpha_1, \ldots, \alpha_q)} \prod_{i=1}^{q} \theta_i^{\alpha_i - 1}, \tag{7}$$

where parameters for each nonterminal are represented by the multinomial distribution $\theta = (\theta_1, \ldots, \theta_q)$ and B is a beta distribution. Since we have no prior knowledge about the distribution of the grammar parameters the rule parameters θ_i and prior weights α_i are uniformly distributed, similar to the original work.[15] The description length of the parameters of the grammar is given by $-\log P(\theta_G|G_S)$.

Second, the structure probability $P(G_S)$ is calculated by directly computing the description length of the structure $DL(G_S)$. $DL(G_S)$ can be defined as the sum of two parts: the description length of the production rule symbols and the description length of number of symbols in the production rule. The description length of the number of symbols is computed from Eq. (8) on the assumption that the length of the production rule is drawn from a Poisson distribution (we use $\mu = 3$) shifted by

one since the smallest possible rule is of length two.

$$-\log P(r-1;\mu) = -\log \frac{e^{-\mu}\mu^{r-1}}{(r-1)!}. \tag{8}$$

Assuming all symbols have the same occurrence probability, we need $\log_2 |\Sigma|$ bits per symbol, where Σ is the set of all symbols. Therefore, the description length of r symbols requires $r \log |\Sigma|$ bits to transmit. The total description length of the structure is given by:

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

Further explanation and justification of the formulation of the description length of the grammar can be found in the original work.[15]

4.3.2. *Description length of the likelihood*

It is not enough to evaluate the description length of the grammar because a grammar chosen purely based on the size will favor a very small grammar which may not explain the data well. The second term in the MDL equation is the description length of the data likelihood $DL(\mathbf{W}|\mathbf{G})$. $DL(\mathbf{W}|\mathbf{G})$ works to balance the effect of the first term by quantifying the expressive power of the grammar.

We first calculate the data likelihood and then convert it into a description length. The data likelihood is calculated using a chart of β probabilities created using the procedure outlined in the original work.[11] The chart defines a function $\beta(N, j, k)$, the probability that the nonterminal N is the root node of a subtree, at abstraction level k, with a terminal substring of length j. Once a chart has been constructed for a sequence $W = \{w_1, \ldots, w_{j_{\max}}\}$, the data likelihood can be computed as a sum of β probabilities for all strings of length $j_{\max}$ produced by the root node S. Due to the insertion of abstraction rules when constructing the initial grammar and the possible creation of abstraction rules at post-processing, the maximum abstraction level $k_{\max}$ is two.

$$P(W_i|\mathbf{G}) = \sum_{k=1}^{k_{\max}} \beta(S, j_{\max}, k). \tag{10}$$

The total likelihood for all the sequences $\mathbf{W}$ is computed by Eq. (11) as a product of likelihoods for each sequence W_i. After the total likelihood has been computed, it is converted into a description length by taking the minus logarithm.

$$P(\mathbf{W}|\mathbf{G}) = \prod_{i=1}^{n} P(W_i|\mathbf{G}). \tag{11}$$

In summary, by calculating the description length of the grammar and the description length of the data likelihood, a framework for evaluating the quality of a presupposition made on the terminal symbols has been created. By identifying the hypothesis grammar that minimizes the total description length, we can find the grammar that optimally describes the data.

4.4. *The recovered grammar*

Our proposed method uses the MDL criterion to discover the most optimal grammar from a set of hypothesis grammars. Here we briefly discuss the nature of the recovered optimal grammar and clarify the focus of our quantitative analysis.

We make no claim that the recovered optimal grammar has a topology that is the same as the original grammar. Except in the special case where the grammar conforms to a set of assumptions made by the COMPRESSIVE algorithm (Sec. 4.2), the underlying assumptions significantly inhibit the type of structures that can be learned.

This however is not a problem for our framework since our aim is to identify a grammar that optimally characterizes the basic structure (rules) between the correct non-noise symbols. To this end, our method is primarily concerned with the relative differences between hypothesis grammars and not the difference from the original grammar. In fact, depending on the form of the original grammar, the basic structures that are learned might be less complex or more complex than the original grammar.

The goal of our quantitative analysis is to show that our method can consistently assign an optimal score to the grammar that uses the correct non-noise symbols and learns the basic structure of the original grammar.

5. Experiments

In this section we explore the conditions under which our proposed method is valid through experiments with synthetic data generated by a known grammar. We also show through an experiment with real data that our method is able to produce intuitive results that align well with our understanding of the target activity.

5.1. *Synthetic data*

The synthetic data for each experiment was created using a stochastic context-free grammar defined according to a set of conditions. A set of d sample strings was generated by the artificial grammar and was used to analyze our proposed method. After the analysis, each hypothesis grammar was ranked according to its description length. Throughout this section we call the grammar which uses the correct non-noise symbols the *true grammar* and use the rank as a measure of the success of our proposed method. We desire for the rank of the true grammar to always be first (i.e. the global solution of the MDL criterion). An example of a predefined grammar is given in Fig. 3 and a ranked list of hypothesis grammars is given in Table 1.

5.1.1. *Inherent insertion noise*

Three different grammar parameters were varied to examine the performance of our method to different types of inherent noise. First, three types of artificial grammars

S	→	EN ACT EX	[1.0]
EN	→	A	[0.5]
EN	→	A INSERT	[0.5]
EX	→	B	[0.5]
EX	→	B INSERT	[0.5]
ACT	→	C	[0.5]
ACT	→	C INSERT	[0.5]
A	→	a	[1.0]
B	→	b	[1.0]
C	→	c	[1.0]
INSERT	→	nd	[0.333]
INSERT	→	ne	[0.333]
INSERT	→	INSERT INSERT	[0.334]

Fig. 3. One pattern grammar with three non-noise symbols and two noise symbols.

Table 1. Ranked list of hypothesis grammars — The true grammar marked in bold is given a suboptimal rank due to a small sized training set.

Rank	Non-Noise Sym.	# Sym.	$DL(G)$	$DL(W\|G)$	Total
1	a b	2	117.85	487.04	604.89
2	a c	2	126.81	489.01	615.82
3	**a b c**	**3**	**348.69**	**327.84**	**676.52**
4	b c	2	187.57	517.32	704.89
5	a	1	85.58	689.34	774.92
6	c	1	89.69	703.08	792.77
7	b	1	113.80	758.57	872.37
8	a ne	2	362.22	622.20	984.42
9	a nd	2	403.36	604.69	1008.05
10		0	70.82	942.46	1013.28
11	nd	1	223.37	826.17	1049.53
12	ne	1	223.37	854.10	1077.46
13	a c ne	3	664.92	415.79	1080.71
14	c nd	2	540.10	566.35	1106.45
15	c ne	2	512.91	604.70	1117.61
16	a c nd	3	749.00	399.35	1148.35
17	a b nd	3	774.74	397.38	1172.12
18	a b ne	3	758.90	422.24	1181.14
19	b nd	2	608.16	636.22	1244.38
20	b ne	2	608.30	675.64	1283.94
21	b c nd	3	981.49	398.73	1380.22
22	b c ne	3	999.61	422.70	1422.31
23	a b nd ne	4	1268.58	260.39	1528.97
24	a b c nd	4	1300.13	257.39	1557.52
25	a b c ne	4	1300.13	257.39	1557.52
26	a c nd ne	4	1300.13	257.39	1557.52
27	b c nd ne	4	1300.13	257.39	1557.52
28	a b c nd ne	5	1300.13	257.39	1557.52
29	nd ne	2	885.68	706.60	1592.28
30	a nd ne	3	1145.23	489.99	1635.22
31	b nd ne	3	1151.62	487.99	1639.61
32	c nd ne	3	1280.75	377.39	1658.15

with different numbers of patterns were defined to evaluate the response of our proposed method to grammars with increasing complexity. Type one grammars had only one basic pattern (one S rule) while type two and type three produced two patterns (two S rules) and three patterns (three S rules), respectively. The basic patterns of type two and type three grammars were different permutations of the same non-noise symbols. An example of a type one grammar and a type two grammar are given in Figs. 3 and 5, respectively. Second, for each type of synthetic grammar, the number of terminal symbols were varied from 6 to 10. Several permutations between the number of noise and non-noise symbols were tested. An insertion noise rule was added for every non-noise production rule to simulate the random insertion of noise between non-noise symbols. Third, to evaluate the effect of the sample size on the results, several training sets consisting of $d = 50, 150, 300, 500, 1000$ randomly produced strings were analyzed for each artificial grammar. The parameters and results for each artificial grammar are given in Table 2.

The results show that our method has identified the correct set of non-noise symbols when the sample size is sufficiently large (Table 2). Equivalently, our method has been shown to produce suboptimal results when the size of the training set was too small. The results also show that complex grammars require more training samples than do simple grammars. It was also observed that the rank of the true grammar converges faster to the top position for simpler grammars (Fig. 4).

Suboptimal results were encountered when the sample size was not sufficient because the learned grammar was under-developed and the data likelihood was under-representative of the data. Specifically with respect to the learned grammar, the insufficient sample size means that the extent of the randomness of the real noise symbols is not fully observed and therefore not fully described by the learned grammar. As a result, grammars using noise symbols are under-developed and are not properly penalized with a long description length.

With respect to the description length of the data likelihood, an insufficient sample size means that the data is not representative of the true set of strings that could be produced by the hidden grammar. As a result, the description length of the data likelihood becomes a small value and is constrained to a narrow range of values. This means that the description length of the data likelihood plays a weaker role in determining the optimal grammar. When these two aspects are combined, a small sample size creates a strong bias toward simple grammars. In fact, in our experiment with synthetic data, the true grammar was always outranked by smaller grammars when the sample size was insufficient (e.g. Table 1). Later, we introduce a strategy for balancing the total description length in Sec. 5.2.3.

5.1.2. *Synthetic system noise*

Despite the fact that the method proposed thus far has been designed to address inherent insertion noise, it has been shown in preliminary experiments that our method is also able to deal with system noise. More specifically, our results show

Table 2. Results with synthetic data (inherent insertion noise).

			Rank of the True Grammar				
Type	Non-Noise	Noise	$d = 50$	$d = 150$	$d = 300$	$d = 500$	$d = 1000$
1	3	3	3	1	1	—	—
1	3	4	3	1	1	—	—
1	3	5	3	1	1	—	—
1	3	6	5	1	1	—	—
1	3	7	4	1	1	—	—
1	4	3	12	4	1	1	1
1	4	4	15	4	1	1	1
1	4	5	11	4	1	1	1
1	4	6	14	4	1	1	1
1	5	3	30	15	5	1	1
1	5	4	34	15	5	1	1
1	5	5	54	15	5	1	1
2	3	3	11	4	1	1	—
2	3	4	12	4	1	1	—
2	3	5	28	4	1	1	—
2	3	6	8	4	1	1	—
2	3	7	30	4	1	1	—
2	4	3	25	11	5	1	1
2	4	4	49	11	5	1	1
2	4	5	28	11	5	1	1
2	4	6	65	13	5	1	1
2	5	3	55	35	16	6	1
2	5	4	91	43	16	6	1
2	5	5	242	34	16	6	1
3	3	3	23	5	1	1	—
3	3	4	28	5	1	1	—
3	3	5	28	6	1	1	—
3	3	6	84	7	1	1	—
3	3	7	177	7	1	1	—
3	4	3	37	26	11	4	1
3	4	4	71	18	11	4	1
3	4	5	102	43	11	4	1
3	4	6	213	89	10	3	1
3	5	3	85	69	26	16	5
3	5	4	87	80	30	16	5
3	5	5	181	136	27	17	5

that our method is able to cope with random insertion, deletion and substitution errors. Insertion caused by system noise introduces the possibility of a non-noise symbol to appear randomly in the input sequence. The deletion of a non-noise symbol creates sequences with incomplete patterns. Substitution is a combination of a deletion and an insertion, where an important non-noise symbols is removed and replaced by either a noise symbol or another non-noise symbol.

One of the grammars used to produce the training samples is given in Fig. 5. In addition to the insertion (INS) rules which represent inherent insertion noise, a substitution (SUB) rule was added to randomly insert a symbol in the place of a

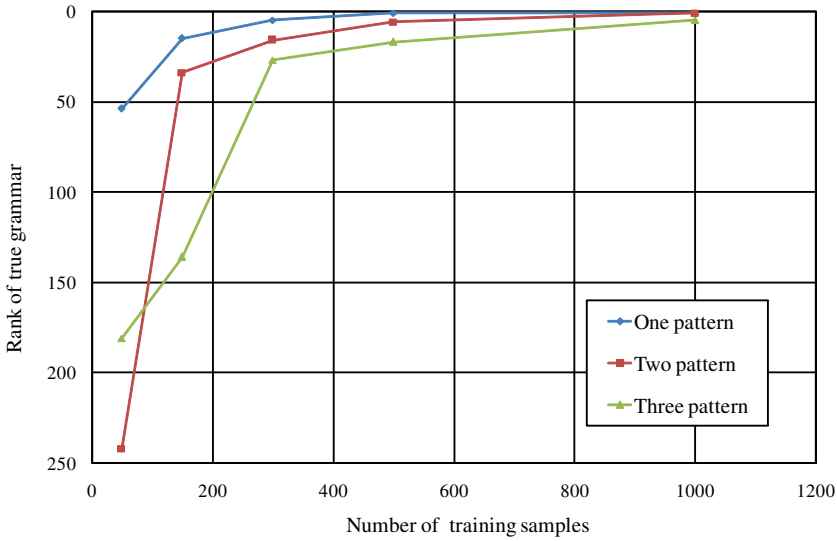

Fig. 4. Rank of the true grammar converging to the top position (for a grammar with five noise symbols and five noise symbols).

S	→	EN	ACT	EX	[0.5]
S	→	ACT	EX	EN	[0.5]
EN	→	A			[0.45]
EN	→	A	INS		[0.45]
EN	→	SUB			[0.10]
EX	→	B			[0.45]
EX	→	B	INS		[0.45]
EX	→	SUB			[0.10]
ACT	→	C			[0.45]
ACT	→	C	INS		[0.45]
ACT	→	SUB			[0.10]
A	→	a			[1.0]
B	→	b			[1.0]
C	→	c			[1.0]
INS	→	nd			[0.25]
INS	→	ne			[0.25]
INS	→	nf			[0.25]
INS	→	INS	INS		[0.25]
SUB	→	nd			[0.30]
SUB	→	ne			[0.30]
SUB	→	nf			[0.30]
SUB	→	A			[0.0333]
SUB	→	B			[0.0333]
SUB	→	C			[0.0334]

Fig. 5. Two pattern grammar with three non-noise symbols and system noise.

non-noise symbol. The parameters of the substitution rules have been distributed in such a way that non-noise symbols are inserted as noise 10% of the time. This is reasonable if we assume that key non-noise symbol detectors have high reliability. Using the artificial grammar, the training data was randomly generated for various values of d.

Table 3. Results with synthetic data (inherent insertion and system noise).

			Rank of the True Grammar				
Type	Non-Noise	Noise	$d = 50$	$d = 150$	$d = 300$	$d = 500$	$d = 1000$
1	3	3	12	3	1	1	1
2	3	3	15	7	4	2	1
3	3	3	23	17	7	4	1

Table 3 shows that the new modes of noise introduced by system noise increased the complexity of the task, which resulted in a need for more training samples to identify the true grammar. Our method was able to recover the correct non-noise symbols despite the increase of noise types because partial patterns could still be described by the CFG while incurring only a minimal increase in grammar size. As a result, the description length of the grammar and the data likelihood of the true grammar attained smaller values relative to those of other hypothesis grammars. These results show that as long as there is more order among the non-noise symbols compared to the noise symbols, an optimal solution can be identified. Consequently, if the structure between the non-noise symbols is corrupted to a degree, such that the randomness of the non-noise symbols becomes similar to the randomness of the noise symbols, our method will only be able to identify a grammar using a subset of the correct set of non-noise symbols as the optimal solution.

5.1.3. *Time complexity*

If we let C be the maximum number of symbols in a single sequence and let B be the number of training samples (sequences), the COMPRESSIVE algorithm has a theoretical time complexity of $O((BC)^2)$ because it makes multiple passes over the input string. However, in practice it is very fast compared to the calculation of the data likelihood when the speed-up techniques introduced in the original work[9] are used. The linear time Sequitur algorithm[8] could also be implemented for additional time savings.

The computation of the beta probabilities in the worst case is $O(PC^DK^D)$, where P is the number of induced productions, K is the maximum number of abstraction levels (for our method $K = 2$) and D is the maximum production length. The beta probabilities must be computed for each sequence, which means the time complexity for computing the data likelihood is $O(BPC^D2^D)$. Furthermore, since our method evaluates every combination of terminal symbols, the total time complexity is $O(2^A(BPC^D2^D))$, where A is the number of terminal symbols. When the hidden grammar is complex, the calculation of the data likelihood dominates the computation time because the average number of symbols in a sequence C and the number of terminal symbols A and the maximum production length D become large. The Stolcke–Earley parser[15] could be implemented as an alternative algorithm to speed up the calculation of the data likelihood.

Fig. 6. Overhead view of the CCD camera mounted above the counter.

5.2. *Experiments with real data*

A surveillance system in a local convenience store was setup to test our proposed method on real data. The system consisted of a single overhead CCD camera (Fig. 6) that captured the hand movements of the employee and the customer. In our experiment a total of more than 9700 frames were recorded and processed offline according to the proposed method. Since the main goal was to learn the high-level grammar (not video segmentation) for a typical employee-customer transaction, the video was manually segmented for each new customer. While we did not address the issue of segmentation in this paper, finding the beginning and ends of an activity will be an important task to address in future works when using a syntactic approach to learning.

5.2.1. *Extracting primitive action symbols*

Primitive actions symbols were detected using simple image processing using application-specific domain knowledge. Skin color was detected in the HSV space by merging a thresholded binary image from each channel. Similarly, the blue tray was detected using different thresholds in the HSV color space. The removal of the scanner and the receipt was detected by monitoring pixel changes over a

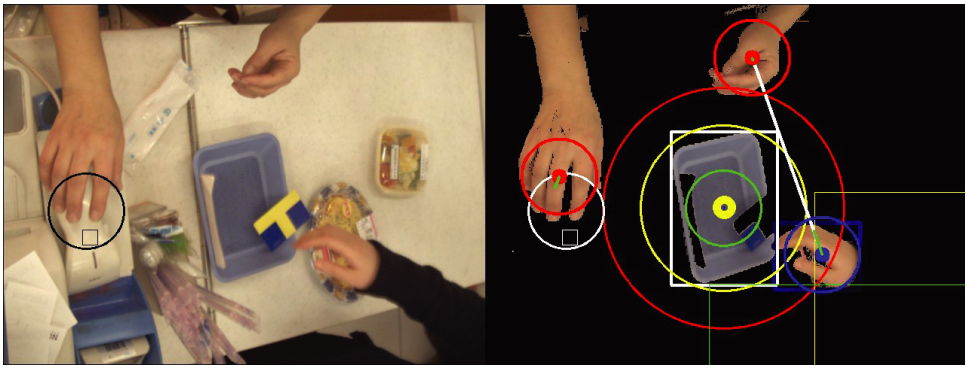

Fig. 7. A frame from the image processing module showing the detection of hands and tray.

small spatio-temporal window over the target region. Similarly, the addition and removal of money into the tray was detected by monitoring a spatio-temporal window over the center of the tray. An example of the results of the image processing module is shown in Fig. 7. For this experiment a total of ten different types of primitive action symbols were extracted. An explanation of the terminals is given in Table 4. We implemented a simple rule-based image processing system to create the primitive action symbols but our method will work with any low-level image processing system that produces a string of primitive actions symbols.

A total of 369 symbols were automatically extracted from the convenience store surveillance video. The longest symbol sequence was 11 symbols long and the shortest sequence was three symbols long. Each sequence was concatenated into one long symbol string as the input to our algorithm. The size of the training data was $d = 55$ strings.

After acquiring the training data, we evaluated each hypothesis for every possible subset of primitive symbols as outlined in Sec. 4.1. Since there were ten

Table 4. Definition of the terminal symbols.

No.	Terminal Symbol	Description
1	CUS_AddedMoney	Money found in tray after customer comes in contact with the tray
2	CUS_MovedTray	Customer moves tray
3	CUS_RemovedMoney	Customer removes money from tray
4	EMP_HandReturns	Employee hand returns after long absence
5	EMP_Interaction	Employee interacts with customer
6	EMP_MovedTray	Employee moves the tray
7	EMP_RemovedMoney	Employee moves money from tray
8	EMP_ReturnedScanner	Employee returns scanner
9	EMP_TookReceipt	Employee takes the receipt from the register
10	EMP_TookScanner	Employee picks up scanner

different terminals symbols, our system evaluated 1024 possible grammars. While our method has the advantage of a complete search over the entire solution space, evaluating every possible combination leads to a combinatorial explosion as the number of terminal symbols increases. We took a brute force approach and evaluated every combination in this experiment but our results suggest that it may be possible to optimize the search by first evaluating grammars that use many non-noise symbols and limit subsequent evaluations to symbol subsets that are contained only in the top scoring set(s). This will be a topic for future work.

5.2.2. *Initial results*

The MDL identifies a single optimal grammar but from a practical perspective, it is useful to present a list of the top hypothetical grammars. The top scoring hypothetical grammar for each class of grammars using the same number of non-noise symbols can be ranked as a list. While a certain user may be satisfied by a grammar that identifies two or three non-noise symbols, another user might desire a more descriptive grammar using five or more non-noise symbols despite the cost of a more complex grammar. Providing such a list would allow the user to choose the preferred grammar from a list of high scoring hypothetical grammars. A list of the top ranking grammars for each class of grammars using the same number of non-noise symbols x is given in Table 5.

We expect to see a global minimum for a hypothetical grammar that uses actions such as *EMP_TookScanner* and *EMP_ReturnScanner* that are known to consistently occur during standard transaction sequences. However, we also know from experiments with synthetic data that a sample size of 55 is likely to produce sub-optimal results when there are more than two true non-noise symbols. In fact, in these initial results a global minimum is found for a grammar that uses only one non-noise symbol *EMP_TookScanner*. As suspected, our method has given more weight to the simplicity of the grammar and less weight to its descriptive ability. Furthermore, we notice that high scores are given to grammars using symbols that occur less frequently in the data. For example, the top scoring grammar using $x = 2$ non-noise symbols includes the terminal *CUS_MovedTray*, an action that was only detected twice in the entire training set. Intuition tells us that general rules should not be generated from symbols of rare occurrence.

5.2.3. *Balancing description lengths*

Figure 8 compares the range (difference between the minimum value and maximum value) of the description lengths of the grammar and the data likelihood produced by the real data. We can see from this figure that the range of the description length of the grammar is consistently greater than the range of the description length of the data likelihood. This indicates that the size of the grammar always has a greater influence on the total description length.

Table 5. Top hypothesis grammars — optimal grammar marked in bold.

x	Non-Noise Symbols	$DL(G)$	$DL(W\|G)$	Total
0		140.11	1319.31	1459.42
1	**EMP_TookScanner**	**221.41**	**1194.29**	**1415.70**
2	CUS_RemovedMoney EMP_TookScanner	245.34	1191.28	1436.62
3	CUS_MovedTray CUS_RemovedMoney EMP_TookScanner	294.19	1187.16	1481.35
4	CUS_MovedTray CUS_RemovedMoney EMP_TookReceipt EMP_TookScanner	493.40	1054.20	1547.60
5	CUS_MovedTray CUS_RemovedMoney EMP_MovedTray EMP_TookReceipt EMP_TookScanner	658.55	1011.17	1669.72
6	CUS_MovedTray CUS_RemovedMoney EMP_MovedTray EMP_ReturnedScanner EMP_TookReceipt EMP_TookScanner	1100.30	818.56	1918.86
7	CUS_MovedTray CUS_RemovedMoney EMP_HandReturns EMP_MovedTray EMP_ReturnedScanner EMP_TookReceipt EMP_TookScanner	1557.83	713.17	2271.00
8	CUS_AddedMoney CUS_MovedTray CUS_RemovedMoney EMP_HandReturns EMP_MovedTray EMP_RemovedMoney EMP_ReturnedScanner EMP_TookScanner	2040.80	545.23	2586.03

As in our case, it may not always be possible to gather enough samples to apply an MDL criterion directly to the training data. In order to compensate for the imbalance between the description length of the grammar and the description length of the data likelihood, it is helpful to introduce a weighting scheme into the MDL criterion.

We can balance the effect of the description length of the grammar and the description length of the data likelihood by introducing a factor γ_x into the MDL

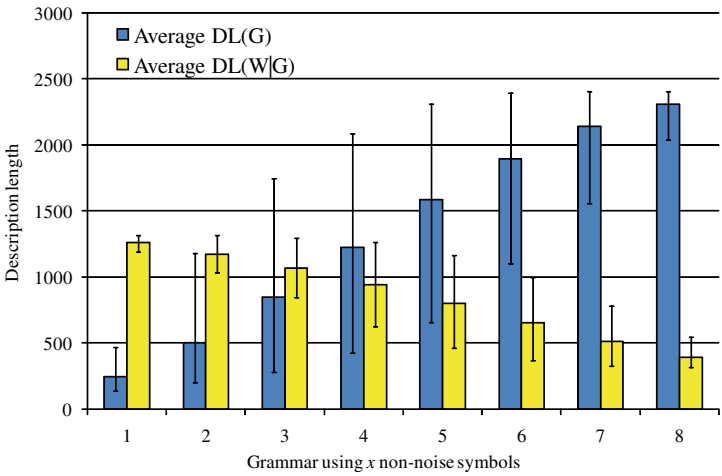

Fig. 8. Imbalance between description lengths — range of the grammar description length is consistently larger than the range of the likelihood description length (error bars show a range of description length).

equation, where γ_x has been interpreted to be the *prior weight* of the grammar or the inverse of the *data multiplier*,[15] or the *representativeness* of the data.[12]

$$\gamma_x DL(\mathbf{G}_x) + DL(\mathbf{W}|\mathbf{G}_x). \tag{12}$$

The value for γ_x is defined as the ratio between the range of the description of the likelihood and the description of the grammar, where x is the number of non-noise symbols used in the grammar. This global prior weighting has the effect of minimizing the contribution of the description length of the grammar and boosts the contribution of the description length of the data likelihood, giving lower priority to grammars that use rare symbols.

$$\gamma_x = \frac{DL_{\max}(\mathbf{W}|\mathbf{G}_x) - DL_{\min}(\mathbf{W}|\mathbf{G}_x)}{DL_{\max}(\mathbf{G}_x) - DL_{\min}(\mathbf{G}_x)}. \tag{13}$$

The top ranking grammar for each class, after compensating for the small size of the training data using our balanced total description length is given in Table 6. Figure 9 shows that the grammar with the smallest overall description length is the hypothesis grammar that uses the three symbols *EMP_ReturnedScanner*, *EMP_TookReceipt* and *EMP_TookScanner*. The grammar learned with these three symbols is given in Fig. 11.

5.2.4. *Recovered basic structure*

The hierarchical structure (parse tree) learned for a common activity H is given in Fig. 10. The parse tree depicts the activity of an employee who first *begins* (node E)

Table 6. Top hypothesis grammars (balanced) — optimal grammar marked in bold.

x	Non-Noise Symbols	γ_x	$\gamma_x DL(G) + DL(W\|G)$
1	EMP_TookScanner	0.383	1279.0016
2	EMP_ReturnedScanner EMP_TookScanner	0.2897	1160.7076
3	**EMP_ReturnedScanner** **EMP_TookReceipt** **EMP_TookScanner**	**0.3096**	**1140.0563**
4	CUS_MovedTray EMP_ReturnedScanner EMP_TookReceipt EMP_TookScanner	0.3847	1211.0414
5	CUS_MovedTray CUS_RemovedMoney EMP_ReturnedScanner EMP_TookReceipt EMP_TookScanner	0.4246	1260.2536
6	CUS_MovedTray CUS_RemovedMoney EMP_MovedTray EMP_ReturnedScanner EMP_TookReceipt EMP_TookScanner	0.4859	1353.1436
7	CUS_AddedMoney CUS_MovedTray CUS_RemovedMoney EMP_MovedTray EMP_RemovedMoney EMP_ReturnedScanner EMP_TookScanner	0.5335	1523.8244
8	CUS_MovedTray EMP_HandReturns EMP_Interaction EMP_MovedTray EMP_RemovedMoney EMP_ReturnedScanner EMP_TookReceipt EMP_TookScanner	0.6228	1784.4875

the transaction by taking the scanner to enter the barcodes of items for purchase into the register. Then, the employee *ends* (node D) the transaction, by returning the scanner to its holder and issuing the receipt.

Notice that the symbols identified as non-noise symbols are all predictable actions performed by the employee. Since the employee has been trained to follow a certain protocol, his actions are predictable and ordered. In contrast, the actions of the customers show less regularity. Therefore, it makes sense that the MDL criterion identifies a grammar dependent only on the predicable actions of the employee as the optimal grammar.

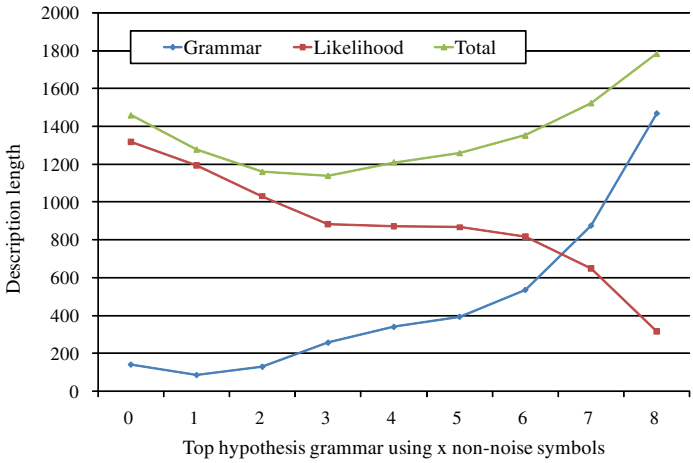

Fig. 9. Description lengths for top hypothesis grammars — global minimum at $x = 3$.

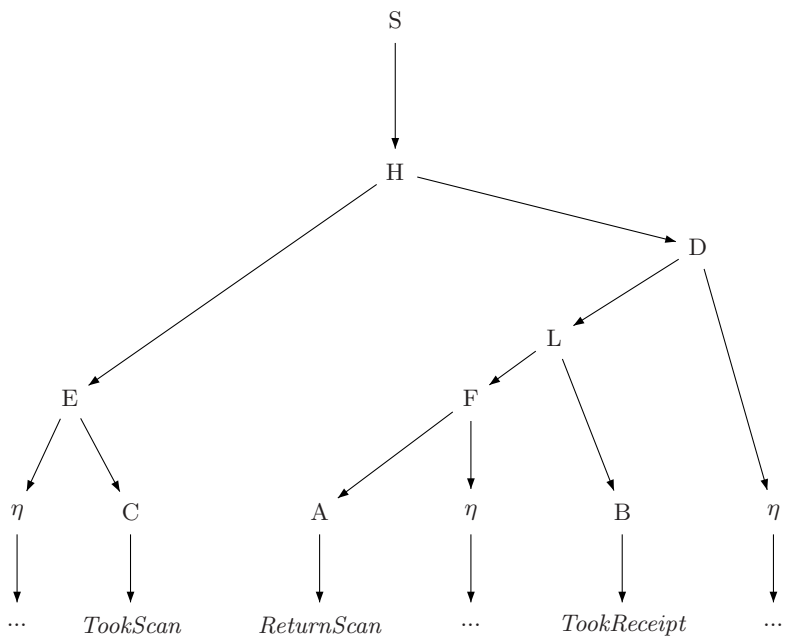

Fig. 10. Parse tree of a common structure found in the training data.

S → D				(0.02)	D → L	η			(1.000)
S → H				(0.16)	E → η	C			(1.000)
S → G				(0.18)	F → A	η			(1.000)
S → N	η			(0.04)	G → C	D			(1.000)
S → J				(0.13)	H → E	D			(1.000)
S → Q				(0.05)	I → $*$	B	η		(1.000)
S → η				(0.02)	J → C	F			(1.000)
S → N				(0.02)	K → $*$	D			(1.000)
S → R				(0.05)	L → F	B			(1.000)
S → J	B			(0.02)	M → C	$*$			(1.000)
S → M	L			(0.04)	N → E	A	B		(1.000)
S → M	A	H		(0.02)	O → E	$*$			(1.000)
S → C	K			(0.04)	P → E	I			(1.000)
S → C	A	M	F	(0.02)	Q → E	K			(1.000)
S → O	F			(0.02)	R → E	L			(1.000)
S → M				(0.02)					
S → O	L			(0.02)	η → η	η			(0.309)
S → O				(0.02)	η → CUS_AddMoney				(0.153)
S → P				(0.05)	η → CUS_MovedTray				(0.006)
S → I				(0.04)	η → CUS_RemMoney				(0.003)
S → K				(0.04)	η → EMP_HandReturn				(0.080)
A → EMP_ReturnedScanner				(1.00)	η → EMP_Interaction				(0.275)
B → EMP_TookReceipt				(1.00)	η → EMP_MovedTray				(0.028)
C → EMP_TookScanner				(1.00)	η → EMP_RemMoney				(0.147)

Fig. 11. Recovered optimal grammar using three non-noise symbols.

6. Conclusion

We have introduced a new method for acquiring the basic structure of an activity from a noisy symbol string produced by video. Our method placed presuppositions on each combination of terminal symbols and tested that hypothesis using an MDL criterion. The MDL equation measured the balance between a compact grammar and a detailed description of the encoded data, and provided a means of quantifying the quality of each presupposition. Experiments with artificial data showed our method is able to correctly identify an optimal grammar when the size of the training data was sufficient. Results also exemplified an inherent bias toward smaller grammars when the size of the training data was insufficient. Based on insights from experimental results, we proposed a way of balancing the MDL equation using a data multiplier γ_x which minimized the bias toward smaller grammars. This new balanced equation resulted in the discovery of a compact grammar that captured the basic structure of activities found in the training data.

While creating a symbol string from video has allowed us to use pre-existing syntactic analysis techniques, we have yet to utilize the full range of the information contained in video. For example, a more intuitive grammar could be attained by analyzing *temporal* information between two actions (e.g. one action always occurs 30 s after another) or by comparing the relative *location* (e.g. two actions occur in the same location) or by observing that two actions are always connected to a *common object*. Future work will use temporal, spatial and contextual information in the grammar learning process.

Furthermore, when we consider the applications of human activity learning techniques, we will in most cases, have some general *a priori* information about the activities to be learned. For example, in our experiments, we already know that

an employee-customer interaction will begin with the placement of an item on the counter and end with a payment for the item. In future works, we will used this type of rough *a priori* grammar to guide our learning process, to discover more subtle and complex grammars found in human activities.

References

1. J. F. Allen, Towards a general theory of action and time, *Artif. Intell.* **23**(2) (1984) 123–154.
2. R. Collins, A. Lipton and T. Kanade, Introduction to the special section on video surveillance, *IEEE Trans. Patt. Anal. Mach. Intell.* **22**(8) (2000) 745–746.
3. N. Ghanem, D. DeMenthon, D. Doermann and L. Davis, Representation and recognition of events in surveillance video using petri nets, *Second IEEE Workshop on Event Mining* (2004), p. 112.
4. Y. A. Ivanov and A. F. Bobick, Recognition of visual activities and interactions by stochastic parsing, *IEEE Trans. Patt. Anal. Mach. Intell.* **22**(8) (2000) 852–872.
5. K. M. Kitani, Y. Sato and A. Sugimoto, Deleted interpolation using a hierarchical Bayesian grammar network for recognizing human activity, *Proc. Second Joint IEEE Int. Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance* (2005), pp. 239–246.
6. D. Minnen, I. A. Essa and T. Starner, Expectation grammars: leveraging high-level expectations for activity recognition, *Proc. IEEE Conf. Computer Vision and Pattern Recognition* (2003), pp. II: 626–632.
7. D. J. Moore and I. A. Essa, Recognizing multitasked activities from video using stochastic context-free grammar, *Proc. Eighteenth Nat. Conf. Artificial Intelligence* (American Association for Artificial Intelligent, 2002), pp. 770–776.
8. C. G. Nevil-Manning and I. H. Witten, Identifying hierarchical structure in sequences: a linear-time algorithm, *J. Artif. Intell. Res. (JAIR)* **7** (1997) 67–82.
9. C. G. Nevil-Manning and I. H. Witten, Online and offline heuristics for inferring hierarchies of repetitions in sequences, *Proc. IEEE* **88**(11) (2000) 1745–1755.
10. A. S. Ogale, A. Karapurkar and Y. Aloimonos, View-invariant modeling and recognition of human actions using grammars, *Proc. Workshop on Dynamical Vision* (2005).
11. D. V. Pynadath and M. P. Wellman, Generalized queries on probabilistic context-free grammars, *IEEE Trans. Patt. Anal. Mach. Intell.* **20**(1) (1998) 65–77.
12. J. R. Quinlan and R. L. Rivest, Inferring decision trees using the minimum description length principle, *Inform. Comput.* **80**(3) (1989) 227–248.
13. M. S. Ryoo and J. K. Aggarwal, Recognition of composite human activities through context-free grammar based representation, *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition* (2006), pp. 1709–1718.
14. Y. Shi, Y. Huang, D. Minnen, A. F. Bobick and I. A. Essa, Propagation networks for recognition of partially ordered sequential action, *Proc. IEEE Conf. Computer Vision and Pattern Recognition* (2004), pp. 862–869.
15. A. Stolcke, *Bayesian Learning of Probabilistic Language Models*. Ph.D. thesis, University of California, Berkeley (1994).
16. T.-S. Wang, H.-Y. Shum, Y.-Q. Xu and N.-N. Zheng, Unsupervised analysis of human gestures, *Proc. IEEE Pacific Rim Conf. Multimedia* **2195** (2001) 174–181.
17. J. M. Zacks and B. Tversky, Event structure in perception and conception, *Psychol. Bull.* **127** (2001) 3–21.

Kris M. Kitani received his B.S. in electrical engineering from the University of Southern California in 1999, and his M.S. in information and communications engineering from the University of Tokyo in 2005.

He is currently a Ph.D. candidate at the University of Tokyo.

Yoichi Sato is an associate professor jointly affiliated with the Graduate School of Interdisciplinary Information Studies, and the Institute of Industrial Science, at the University of Tokyo, Japan. He received the BSE degree from the University of Tokyo in 1990, and the M.S. and Ph.D. degrees in robotics from the School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, in 1993 and 1997 respectively.

His research interests include physics-based vision, reflectance analysis, image-based modeling and rendering, tracking and gesture analysis, and computer vision for HCI.

Akihiro Sugimoto received his B.Sc., M.Sc., and D.Eng. degrees in mathematical engineering from the University of Tokyo in 1987, 1989, and 1996, respectively. After working at Hitachi Advanced Research Laboratory, ATR, and Kyoto University, he joined the National Institute of Informatics, Japan, where he is currently a professor. From 2006 to 2007, he was a visiting professor at ESIEE, France. He received a Paper Award from the Information Processing Society in 2001.

He is a member of IEEE. He is interested in mathematical methods in engineering.

In particular, his current main research interests include discrete mathematics, approximation algorithm, vision geometry, and modeling of human vision.