Recovering the Basic Structure of Human Activities from a Video-Based Symbol String

Kris M. Kitani, Yoichi Sato
The University of Tokyo
Institute of Industrial Science
Tokyo 158-8505, Japan
{kitani, ysato}@iis.u-tokyo.ac.jp

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

Abstract

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 an experiment with real video from a local convenience store.

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 ([2], [10], [5], [4], [7]). 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 [11], this hierarchical structure is also characteristic of the primitive actions of a human activity [1] and like sentences, activities are perceived 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 non-hierarchical finite-state machines (finite-state automata, hidden Markov models, *n*-grams, etc.) have been used for human activity recognition but are limited by the fact that they cannot 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 hand-crafted their own grammars and have avoided the issue of grammar learning. Ivanov [2] 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 it's probabilities however, were defined by Ivanov. Moore[5] used an 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 [4] 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 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 [10] used an experimental scenario similar to Ivanov and implemented HMMs to produce primitive action symbols from a video segment of a conductors hand motions. The primitive actions produced by the HMMs were then fed into a pre-existing CFG learning algorithm COMPRESSIVE [6] 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 errors in the input symbol string. That is, an unstable detection would have a very adverse affect 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 produced by stochastic detectors on a noisy video signal created by human actors.

In summary, most of the works using CFGs for activity analysis have used grammars manually designed by knowl-

[1] We use the term *activity* based on terminology introduced in [1] instead of the term *event* in [11] to refer to the high-level description of a temporal sequence of primitive actions.

edge 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 a class of the best set of terminal symbols that yields the most compact and descriptive activity grammars.

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 randomly remove all `y` symbols from the stream making the *presupposition* that they are noise symbols [2]. 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 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 assumed *a priori* to be noise, we were able to obtain a compact grammar ($\texttt{A} \rightarrow \texttt{a b c}$) and a deterministic description of the basic structure of the original symbol string $\texttt{S} \rightarrow \texttt{A A A}$. Thus we reason that it is highly probable that `x` and `y` are in fact noise symbols.

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

3 Proposed method

It is necessary to first understand the focus of our proposed approach before we proceed to explain its details. In general, learning an activity grammar consists of two parts: (1) low-level image processing to extract the primitive symbols and (2) high-level syntactic analysis to learn the grammar from the primitive symbol string. The primary focus of our approach is on high-level grammar learning and therefore we provisionally set aside the equally important topic of low-level image processing. That is, we make the assumption that we already have a relatively reliable (but noisy) low-level image processing system to produce the primitive symbol string.

In this section we formalize the key concepts introduced through the example in section 2 and explain how we perform high-level grammar learning. We begin by defining the characteristics of noise symbols and then proceed to explain how our proposed method is able to discover basic activity structures from noisy video data using the MDL principle.

3.1 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 hide the basic structure of the activity that we want to learn. First there 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) certain 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. The second 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 tend to appear with irregular frequency and order, and fill in the gaps between the important action symbols.

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 sub-problem, namely, insertion noise in the training data. First, we make the assertion that a symbols is either a (1) noise symbol or a (2) non-noise symbol. 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.

3.2 Setting up the presuppositions

Given a set of noisy training data (strings of primitive action symbols from video), we need a means of identifying the noise symbols to ensure that we use only non-noise symbols to recover an activity grammar. In order to identify the noise symbols, various presuppositions are set against the primitive symbols and those presuppositions are later evaluated using an MDL criterion. Here we explain how the initial presuppositions are set up.

Since we do not know *a priori* which symbols are noise, we need to evaluate every possible presupposition against each type of primitive symbol. For example, if there are two types of primitive symbols $\mathtt{x}$ and $\mathtt{y}$, the following four presuppositions are possible: $\{\emptyset\}, \{x\}, \{y\}, \{x, y\}$ (None of the primitive symbols are noise, x is noise, y is noise or x and y are noise, respectively).

For a given presupposition made on the primitive symbols, an initial grammar is constructed to reflect that presupposition. First we create a set of production rules $N_i \rightarrow w_i^+$ for each presupposed non-noise symbol, where w_i^+ 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. Second, we create a set of generic preterminal production rules for presupposed noise symbols in the form $* \rightarrow w_i^-$, where w_i^- is a noise terminal symbol and 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. Third, we create a new rule that contains the whole input symbol string $\mathbf{W}'$ encoded by the presupposition. An example of setting up a presupposition is given in figure 1.

To attain the encoded input symbol string $\mathbf{W}'$ we begin with the plain input symbol string $\mathbf{W} = \{W_1 \dots W_j\}$, which is a series of concatenated activity sequences, where each sequence W_i is a string of primitive symbols headed by a start marker $\epsilon_i w_1 \dots w_k$. We encode the plain input string to reflect the presuppositions made about the symbols by replacing each terminal symbol w with the appropriate nonterminal symbol, using the preterminal productions rules created earlier. After all of the new rules have been inserted into the grammar, we obtain the set of initial rules $\mathbf{R}_0$.

$$\mathbf{R}_0 = \left\{ \begin{array}{ll} S \rightarrow \mathbf{W}', & * \rightarrow **, \\ N_1 \rightarrow w_1^+, & * \rightarrow w_1^-, \\ \dots & \dots \\ N_u \rightarrow w_u^+, & * \rightarrow w_v^- \end{array} \right\}.$$

Input strings:
$W_1 = 1cabaab$
$W_2 = 2abacab$
$W_3 = 3abaabc$

Presupposition:
c is noise.

Initial grammar:
$S \rightarrow 1 * ABAAB2ABA * AB3ABAAB*$
$A \rightarrow a$
$B \rightarrow b$
$* \rightarrow c$
$* \rightarrow **$

Figure 1. Setting up a presupposition.

3.3 Learning the hypothetical grammar

Now that we have effectively encoded the presuppositions on the primitive action symbols into the initial grammar, we proceed to learn the hypothetical grammar. The heuristic CFG learning algorithm COMPRESSIVE is implemented here to learn the grammar and the occurrence counts for new rules are stored while the algorithm is running. 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:

$$\max_{\nu} \Delta DL = m \cdot n - (m + 1) - n. \qquad (1)$$

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. 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 encoded yet again to produce $\mathbf{W}''$. In the post-processing step, we group segments in $\mathbf{W}''$ that have the same structure. To do this, we first remove the S rule, $S \rightarrow W_1'' \cdots W_m''$, from the grammar. Next, we separate each sequence and create a new S rule for each sequence: $S \rightarrow W_1'', \cdots, S \rightarrow W_m''$. These

new rules are then inserted back into the grammar, where multiple occurrences of the same encoded sequence are inserted only once, whereas counts are continually updated. The production rule probabilities are calculated with the follow equation:

$$P(A \rightarrow \lambda_i^*) = \frac{c(A \rightarrow \lambda_i^*)}{\sum_j c(A \rightarrow \lambda_j^*)}, \quad (2)$$

such that A is a nonterminal and λ^* is the right-hand side of the rule. Rules with zero probability are removed from the grammar.

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

3.4 Testing using the MDL principle

We want 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 $\mathbf{W}$ encoded by the grammar (data likelihood) $DL(\mathbf{W}|\mathbf{G})$.

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

In this section, we use the encoding technique proposed in [9] to find the description length of the grammar and we use inside probabilities to calculate the description length of the data likelihood.

3.4.1 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,

$$P(\mathbf{G}) = P(G_S, \theta_G) = P(\theta_G|G_S)P(G_S), \quad (4)$$

the description length of the grammar can be acquired by summing 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 grammar parameter probability $P(\theta_G|G_S)$ is calculated as the product of Dirichlet distributions (equation 5), such that each Dirichlet distribution represents an equally distributed probability across all n possible productions of a nonterminal symbol N.

$$P_N(\theta_G|G_S) = \frac{1}{B(\alpha_1, \ldots, \alpha_n)} \prod_{i=1}^{n} \theta_i^{\alpha_i - 1}, \quad (5)$$

where parameters for each nonterminal is represented by the multinomial distribution $\theta = (\theta_1, \ldots, \theta_n)$ and B is a beta distribution. Each rule has an equality distributed probability θ_i and the equality distributed prior weights α_i conform to the conditions $\sum_{i=1}^{n} \alpha_i = 1$ and $\alpha_i < 1$. 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 length of the production rule $code(k)$ and (2) the symbols of the production rule $code(s)$. $code(k)$ is computed from equation (6) on the assumption that the length of the production rule is drawn from a Poisson distribution (we use $\eta = 3$) shifted by one since the smallest possible rule is of length two.

$$-\log p(k-1;\eta) = -\log \frac{e^{-\eta}\eta^{k-1}}{(k-1)!}. \quad (6)$$

Assuming all symbols have the same occurrence probability, we need $\log_2 |\Sigma|$ bits per symbol, such that Σ is the set of all symbols. Therefore, $code(s)$ of a rule with k symbols requires $k \log |\Sigma|$ bits to describe. The description length of the structure is given by:

$$DL(G_S) = \sum_{R \in \mathbf{R}} \left(-\log p(k-1;\eta) + k \log |\Sigma|\right). \quad (7)$$

3.4.2 Description length of the likelihood

It is not enough to evaluate the description length of the grammar because a grammar chosen purely based on grammar 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.

In our method, we first calculate the data likelihood and then convert it into a description length. That is, we use a chart of inside probabilities to calculate the likelihood using the initialization equation (8, 9) and the recursive equation (10), 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

Figure 2. Overhead view of the CCD camera mounted above the counter.

Table 1. 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

of $\{j_1, \ldots, j_m\}$ that sums to j. k_i is two when the symbol is a preterminal symbol and one otherwise.

$$\beta(T, i, 1, 1) = 1.0 \quad (8)$$

$$\beta(N, i, 1, 2) = P(N \rightarrow T) \cdot \beta(T, i, 1, 1) \quad (9)$$

$$\beta(N, i, j, k) = P(N \rightarrow N_1 \cdots N_m) \cdot \sum_{m P_m} \beta(N_1, i_1, j_1, k_1) + \cdots + \beta(N_m, i_m, j_m, k_m) \quad (10)$$

The likelihood for one sequence W_i is calculated from equation (11), as the sum of all S inside probabilities that start at index one. The total likelihood for all the sequences $\mathbf{W}$ is computed by equation (12). After the total likelihood has been computed, it is converted into a description length by taking the minus logarithm.

$$P(W_i|\mathbf{G}) = \sum_{k=1}^{2} \beta(S, 1, j_{max}, k), \quad (11)$$

$$P(\mathbf{W}|\mathbf{G}) = \prod_{i=1}^{n} P(W_i|\mathbf{G}). \quad (12)$$

In summary, by calculating the description length of the grammar and the description length of the data likelihood, we have evaluated the quality of the presuppositions made on the terminal symbols.

4 Experimental results

We implemented a surveillance system in a local convenience store to test our method on real data. The system consisted of a single overhead CCD camera (images shown in figure 2) that captured the hand movements of the employee and the customer. In our experiment a total of 9700 plus frames were recorded and processed offline according to our proposed method. Since our 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 (punctuations) of an activity will be an important task to address in future works when using a syntactic approach to learning.

To produce the training data (action symbol string), a low-level image processing unit was designed to detect various objects (hands, tray, money, etc.) and output ten different types of terminal symbols based on a set of heuristic rules. An explanation of the terminals is given in table 1. Although we implemented a simple rule-based image processing system to create the primitive action symbols, our method will work with any low-level image processing system, as long as it is able to produce a string of primitive actions symbols. As for our system, a total of 369 symbols (not including the start markers ϵ) were automatically produced from the convenience store video. The longest symbol sequence was eleven symbols long and the shortest sequence was three symbols long. Each sequence was concatenated into one long symbol string as an input to our algorithm.

After acquiring the training data, we evaluated each presupposition for every possible subset of primitive symbols as outlined in 3.2. Since there were ten different terminals symbols, our system evaluated 1024 possible grammars. While our method has the advantage of covering the en-

tire solution space, evaluating every possible combination leads to a combinatorial explosion as the number of terminal symbols increase. Although we took a brute force approach and evaluated every combination in this experiment, our results suggest that it is 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).

We present the results of our approach from two different perspectives. First from a practical perspective, it is useful to present a list of the top candidate grammars with respect to the number of terminals symbols that were used as non-noise symbols (table 2). For example, one user may be satisfied by a compact grammar that identifies two or three non-noise symbols, while another user may seek a more descriptive grammar using ten or more non-noise symbols. The user should be able to choose the best grammar based on the number of non-noise symbols utilized.

Second, from the perspective of evaluation, it may be necessary to identify one best grammar. In general the system will favor grammars that use fewer non-noise symbols because they have shorter description lengths. To promote more equality among the grammars, we divide the total description length by the number of terminal symbols used as non-noise symbols in the grammar. This results in a more balanced evaluation that gives more priority to grammars that use more non-noise symbols. This is a method used in sentence compression and speech recognition [3]. A graph of the normalized scores of the top six grammars are given in figure 3. Notice how the U-shaped plot reveals that the grammar using six symbols has the smallest (best) adjusted score.

Table 2. Top candidate grammars (Biased).

No.	Non-noise Symbols	DL(G)	DL(W|G)	DL(G|W)
1	EMP_TookScanner	221.41	1194.29	1415.7
2	CUS_RemovedMoney EMP_TookScanner	245.339	1191.28	1436.619
3	CUS_MovedTray CUS_RemovedMoney EMP_TookScanner	294.193	1187.16	1481.353
4	CUS_MovedTray CUS_RemovedMoney EMP_TookReceipt EMP_TookScanner	493.4	1054.2	1547.6
5	CUS_MovedTray CUS_RemovedMoney EMP_MovedTray EMP_TookReceipt EMP_TookScanner	658.548	1011.17	1669.718
6	**CUS_MovedTray** **CUS_RemovedMoney** **EMP_MovedTray** **EMP_ReturnedScanner** **EMP_TookReceipt** **EMP_TookScanner**	**1100.3**	**818.556**	**1918.856**
7	CUS_MovedTray CUS_RemovedMoney EMP_HandReturns EMP_MovedTray EMP_ReturnedScanner EMP_TookReceipt EMP_TookScanner	1557.83	713.169	2270.999
8	CUS_AddedMoney CUS_MovedTray CUS_RemovedMoney EMP_HandReturns EMP_MovedTray EMP_RemovedMoney EMP_ReturnedScanner EMP_TookScanner	2040.8	545.225	2586.025

5 Balancing description lengths

As alluded to in the previous section, the MDL criteria displays a bias toward smaller grammars. To be precise, the formulation of the description length of the grammar assigns values on a range greater than the range of the description length of the data likelihood, which causes the grammar size to have more influence on the total description length. This phenomenon was observed in tests with artificial data and was also found to be true in our experiments with real data (figure 4).

This imbalance between description lengths causes grammars that use rare symbols (smaller grammar) to be given better total scores while grammars that use frequently occurring symbols (bigger grammar) are given bad total scores. We see this principle at work in our results. For example, the terminal *CUS_MovedTray* was only detected once in two out of 55 data sequences but it is included in the top candidate grammar that uses three symbols (see table 2). Intuition tells us that rules should not be made from symbols that occur only once or twice in the training data. This intuition is actually represented by the description length of the data likelihood but as we stated before it is overpowered by the description length of the grammar.

We can balance the effect of the description length of the grammar and the description length of the data likelihood by introducing a factor α into the MDL equation, where α has been interpreted to be the *prior weight* of the grammar or the inverse of the *data multiplier* [9], or the *representativeness* of the data [8].

$$DL(\mathbf{G}|\mathbf{W}) = \alpha DL(\mathbf{G}) + DL(\mathbf{W}|\mathbf{G}). \qquad (13)$$

We set a value for α as the ratio between the range of the description of the grammar and the description of the likelihood. This 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.

The top candidate grammars relative to the number of non-noise symbols used is given in table 3. Notice now that terminal symbols with consistent occurrence are selected first. The grammar with the smallest overall description length is the candidate grammar that uses the

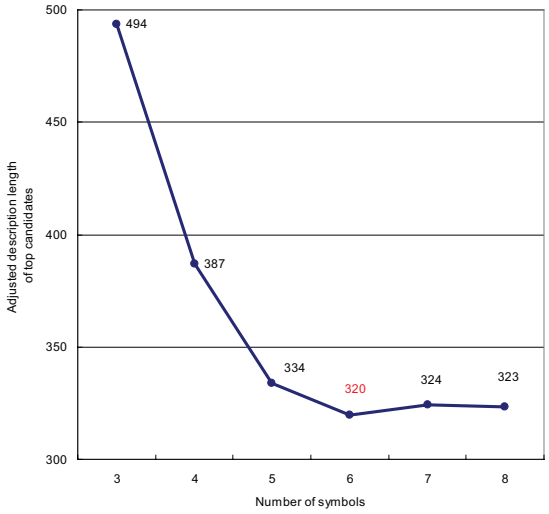

Figure 3. Normalized description lengths for top six candidate grammars.

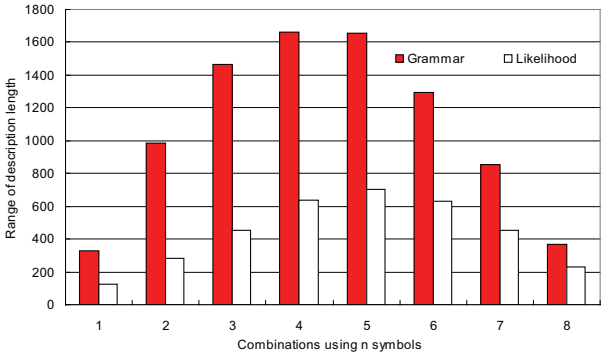

Figure 4. Imbalance between the range of the description lengths.

three symbols *EMP_ReturnedScanner*, *EMP_TookReceipt* and *EMP_TookScanner* with a description length of 1141 bits. The grammar learned with these three symbols is given in table 4.

The hierarchical structure (parse tree) learned for a common activity H is given in figure 5. The parse tree depicts the activity of an employee who first *begins* (node E) 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.

Table 3. Top candidate grammars (Balanced).

No.	Non-noise Symbols	α	αDL(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

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 presupposition 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. Results showed that the formulation of the description lengths created an inherent bias toward smaller grammars and causes unintuitive results. Based on insights from initial results, we proposed a way of balancing the MDL equation using the data multiplier α which minimized the bias toward smaller grammars. This new balanced equation resulted in the discovery of a detailed description of the data likelihood and 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 at-

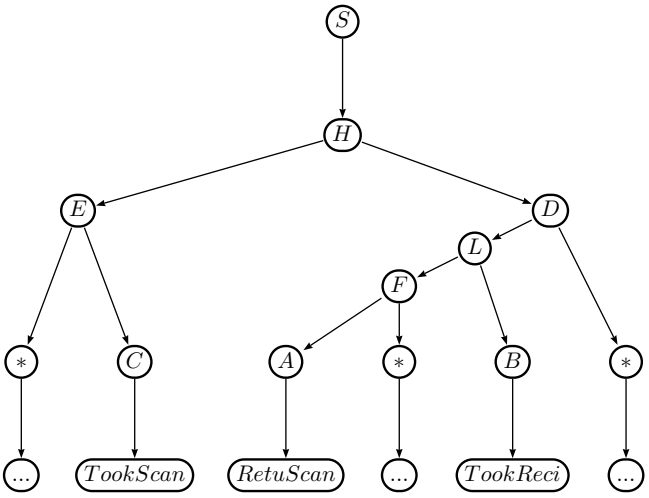

Figure 5. Parse tree of a common structure found in the training data.

Table 4. Learned grammar using three non-noise symbols.

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

tained by analyzing *temporal* information between two actions (e.g. one action always occurs 30 seconds 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] R. Collins, A. Lipton, and T. Kanade. Introduction to the Special Section on Video Surveillance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):745–746, 2000.

[2] 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.

[3] K. Knight and D. Marcu. Statistics-based summarization - step one: Sentence compression. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 703–710. AAAI Press / The MIT Press, 2000.

[4] 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.

[5] 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.

[6] 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.

[7] A. S. Ogale, A. Karapurkar, and Y. Aloimonos. View-Invariant Modeling and Recognition of Human Actions Using Grammars. In *Proceedings of the Workshop on Dynamical Vision*, 2005.

[8] J. R. Quinlan and R. L. Rivest. Inferring Decision Trees Using the Minimum Description Length Principle. *Information and Computation*, 80(3):227–248, 1989.

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

[10] 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.

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