M.I.T Media Laboratory Perceptual Computing Section Technical Report No. 449
Submitted to: IEEE CVPR'98

Action Recognition using Probabilistic Parsing

A. F. Bobick and Y. A. Ivanov
Room E15-383, The Media Laboratory
Massachusetts Institute of Technology
20 Ames St., Cambridge, MA 02139

Abstract

A new approach to the recognition of temporal behaviors and activities is presented. The fundamental idea, inspired by work in speech recognition, is to divide the inference problem into two levels. The lower level is performed using standard independent probabilistic temporal event detectors such as hidden Markov models (HMMs) to propose candidate detections of low level temporal features. The outputs of these detectors provide the input stream for a stochastic context-free grammar parsing mechanism. The grammar and parser provide longer range temporal constraints, disambiguate uncertain low level detections, and allow the inclusion of a priori knowledge about the structure of temporal events in a given domain. To achieve such a system we provide techniques for generating a discrete symbol stream from continuous low level detectors, for enforcing temporal exclusion constraints during parsing, and for generating a control method for low level feature application based upon the current parsing state. We demonstrate the approach in several experiments using both visual and other sensing data.

1 Intoduction: stochastic action recognition

In the last several years there has been a tremendous growth in the amount of computer vision research aimed at understanding *action*. As noted by Bobick [2] these efforts have ranged from the interpretation of basic movements such as recognizing someone walking or sitting, to the more abstract task of providing a Newtonian physics description of of the motion of several objects.

In particular, there has been emphasis on activities or behaviors where the entity to be recognized may be considered as a stochastically predictable sequence of states. The greatest number of examples come form work in gesture recognition [14, 1, 13] where analogies to speech and handwriting recognition inspired researchers to devise *hidden Markov model* methods for the classification of gestures. The basic premise of the approach is that the visual phenomena observed can be considered Markovian in some feature space, and that sufficient training data exists to automatically learn a suitable model to characterize the data.

Our research interests lie in the area of vision where observations span extended periods of time. We often find ourselves in the situations where purely statistical approaches to recognition are less than ideal. These situations can be characterized by one or more of the following properties:

- complete data sets are not always available, but smaller examples could easily be found;
- semantically equivalent processes possess radically different statistical properties;
- competing hypotheses can absorb different lengths of the input stream raising the need for naturally supported temporal segmentation;
- structure of the process is difficult to learn but is explicit and a priori known.

Consider a simple example - we can draw a square with a hand in the air in either clockwise or counterclockwise direction. In either case our "square" model should indicate that the square is being drawn. This seemingly simple task requires significant effort using only the statistical pattern recognition techniques.

The human observer, on the other hand, can provide a set of useful heuristics for a system which would model the human's higher level perception. As we recognize the need to characterize a signal by these heuristics, we turn our attention to syntactic pattern recognition and combined statistical-syntactic approaches, which would allow us to address the problems listed above.

To take advantage of these techniques, we divide the activity recognition problem into two components. The lower level is performed using standard independent probabilistic temporal event detectors such as HMMs to propose candidate detections of low level temporal features. The outputs of these detectors provide the input stream for a stochastic context-free grammar parsing mechanism. The grammar and parser enforce longer range temporal constraints, disambiguate or correct uncertain or mislabeled low level detections, and allow the inclusion of a priori knowledge about the structure of temporal events in a given domain.

For many domains such a division is clear. For example, consider ballroom dancing. There are a small number of primitives (e.g. `right-leg-back`) which are then structured into higher level units (e.g. `box-step`, `quarter-turn`, etc.). Typically one will have many examples of `right-leg-back` drawn from the relatively few examples each of the higher level behaviors. Another example might be recognizing a car executing a parallel parking maneuver. The higher level activity can be described as first a car executes an `pull-along-side` primitive followed by an arbitrary number of cycles through the pattern `turn-wheels-left`, `back-up`, `turn-wheels-right`, `pull-forward`. In these instances, there is a natural division between atomic, statistically abundant primitives and higher level coordinated behavior.

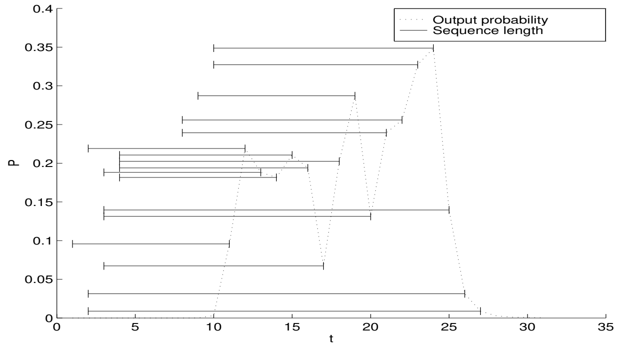

Figure 1: HMM output plot. Each point of the probability plot is the normalized maximum likelihood of the HMM at the current sample of the input signal. This value is found as a result of the backwards search on the FIFO queue, and corresponds to a sequence of a certain length. The plot shows those lengths as horizontal lines ending at the corresponding sample of the probability plot.

2 Related work

Vast amount of work in syntactic pattern recognition has been devoted to the areas of image and speech recognition. A review of syntactic pattern recognition and related methods can be found in [12].

There are many examples of attempts to enforce semantic constraints in recognition of visual data. For instance, Courtney ([4]) uses structural approach to interpreting action in surveillance setting. Grammatical approach to visual data recognition was used by Brand ([3]), who uses a simple non-stochastic grammar to recognize sequences of discrete events. Both authors formulate fairly universal events, common to monitoring applications, based on blob interaction primitives. Using combined probabilistic-syntactic approaches to problems of vision is shown in [16] and [7].

Examples of probabilistic parsing in speech processing can be found in literature (eg. [9, 6], etc.). In particular, [15] developing probabilistic extensions to Earley context-free parser is of interest.

3 Probabilistic input formation

To recognize the components of the model vocabulary, we train one HMM per atomic gesture. At run-time each of these HMMs performs a Viterbi parse ([10]) of the incoming signal and computes the likelihood of the gesture primitive. The run-time algorithm used by the HMMs to recognize the words of the gesture vocabulary is a version of [5] which performs a "backward" match of the signal over a window of a reasonable size. At each time step the algorithm outputs the estimated likelihood of the sequence which ends at the current sample as well as the length of this sequence. We will later exploit this property to enforce *temporal consistency*: only one low level temporal feature is happening at any one time.

All the HMMs are run in parallel, providing the parser with maximum normalized probability and the corresponding sequence length at each time step.[1] Figure 1 shows the output of a single HMM in a bank, illustrating the relation between output probabilities and sequence lengths.

Event generation

The continuous vector output of the HMM bank now has to be represented by a series of "events" which we want to consider for probabilistic parsing. It is these events, in terms of which we will express our action grammar. In this step, we do not need to make strong decisions about discarding any events - we just need to generate a sufficient stream for the probabilistic parser such that it can perform structural rectification of these "suggestions" and find an interpretation which is structurally consistent (i.e. grammatical), is temporally consistent (uses non-overlapping sequence of primitives), and which has maximum likelihood given the outputs of the low level feature detectors (HMMs).

For the tasks discussed here, a simple discretization procedure provides good results. For each HMM in the bank we select a very small threshold to cut off the noise in the output, and then search each of the areas of non-zero probabilities for a maximum. Figure 2a shows an example of the output of the HMM bank, superimposed with the results of discretization.

After discretization we replace the continuous time vector output of the HMM bank with the discrete symbol stream generated by discarding any interval of the discretized signal in which all values are zero. Figure 2b displays an example of the resulting event sequence generated.

4 Probabilistic parsing

Our goal in parsing the sequence from figure 2b is to find a path through this input which is a) consistent with the overall expected structure, b) forms a temporally consistent sequence and c) has an overall maximum probability.

To perform probabilistic parsing of such a sequence we use an efficient Earley-based context-free parsing algorithm extended for probabilistic input ([15]).

The structure of the sequence is described by a Stochastic Context-Free Grammar (SCFG), which is formed by associating a probability with each production. Analogously to HMMs, forward, backward and inner probabilities are introduced. A forward probability, for instance, is computed as a sum of probabilities of all the derivations which form a prefix $w_1 \dots w_t$ of the string $w_1 \dots w_T$.

Probability of a path is computed by a process which iterates over three basic steps - *scanning*, *completion* and *prediction*. After initialization, *scanning* reads a symbol from the input stream and matches it with the initial set of rules. The rules which do not satisfy the symbol read from the stream and, hence, corresponding pending derivations get pruned from the parse tree. *Completion* step, given a set of productions which just have been confirmed by scanning, updates the positions in all the pending derivations all the way up the derivation tree. *Prediction* traverses the grammar tree all the way down to the leaf nodes and

[1] Alternatively, we can search for the position in the trellis where the product of the probability and the sequence length is maximal. This will result in finding the weighted maximum, which corresponds to the maximum probability of the sequence of the maximum length.

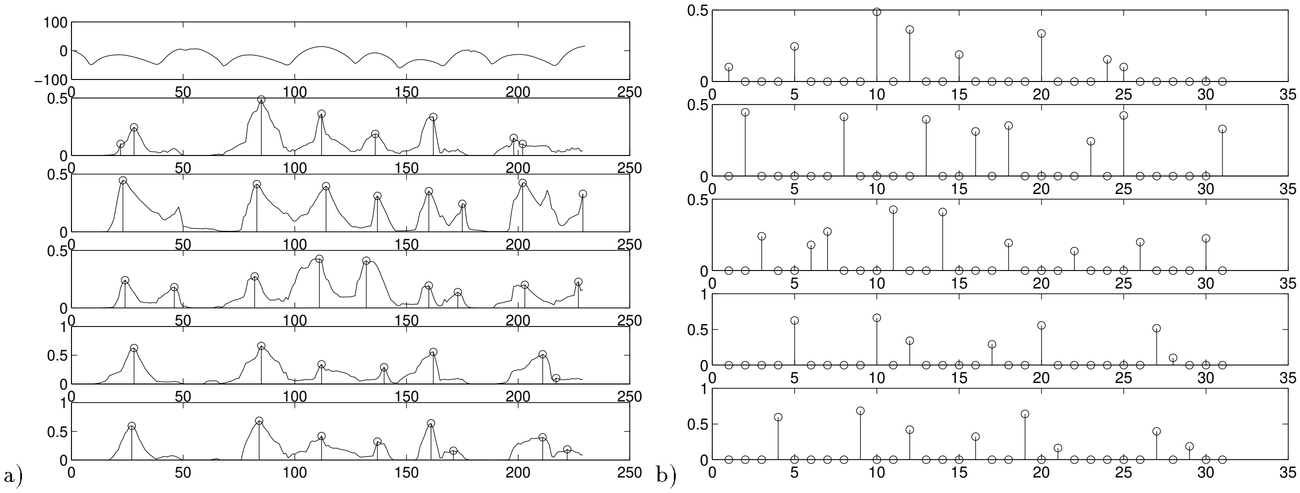

Figure 2: Example of the output of the HMM bank. a) Top plot shows the input sequence. Five plots below it show output of each HMM in the bank superimposed with the result of discretization. b) Corresponding discretized "compressed" sequence. Input vectors for the probabilistic parser are formed by taking the samples of all sequences for each time step.

finds a set of production rules which are currently possible given the position in the input and in the parse tree. Then the scanning is applied again. This combined top-to-bottom and bottom-up approach provides effective means for propagating the probabilities through the parsing process.

A Viterbi path is traced during parsing as a single derivation path with maximum path probability.

4.1 Notation and algorithms

The notion of a *State*, is an important part of the Earley parsing algorithm. A state is denoted as:

$$i : X_k \rightarrow \lambda.Y\mu$$

where '**.**' is the marker of the current position in the corresponding production, i is the position in the input stream, and k is the starting index of the substring to which the above production is being applied. Nonterminal X is said to dominate substring $w_k ... w_i ... w_l$, where, in the case of the above state, w_l is the last terminal of substring μ. The parser generates a *state set* for each position in the input. This also means that in our notation all the states with the same index i belong to the same i-th state set corresponding to i-th position in the input stream. A state "explains" a string which it dominates giving a possible interpretation of symbols $w_k ... w_i$.

In our further discussion we will use expressions $A \Rightarrow B$ in context of state generation to read "A generates B".

Scanning

Scanning step simply reads the input symbol and matches it against all pending states for the next iteration:

$$i : X_k \rightarrow \lambda.a\mu \quad [\alpha, \gamma] \quad \Longrightarrow i+1 : X_k \rightarrow \lambda a.\mu \quad [\alpha, \gamma]$$

where α and γ are forward and inner probabilities.

Note the increase in i index. This signifies the fact that scanning step inserts the states into the new state set for the next iteration of the algorithm.

Completion

Completion uses the results of scanning to advance positions in the parse tree. In its simplified probabilistic form [2], completion step generates following states:

$$\left\{ \begin{array}{l} j : X_k \rightarrow \lambda.Y\mu \quad [\alpha, \gamma] \\ i : Y_j \rightarrow \nu. \quad [\alpha'', \gamma''] \end{array} \right. \Longrightarrow i : X_k \rightarrow \lambda Y.\mu \quad [\alpha', \gamma']$$

$$\begin{array}{rcl} \alpha' & = & \sum_{\forall \lambda, \mu} \alpha(i : X_k \rightarrow \lambda.Y\mu)\gamma''(i : Y_j \rightarrow \nu.) \\ \gamma' & = & \sum_{\forall \lambda, \mu} \gamma(i : X_k \rightarrow \lambda.Y\mu)\gamma''(i : Y_j \rightarrow \nu.) \end{array}$$

Prediction

Prediction step is used to hypothesize the possible continuation of the input based on current position in the parse tree:

$$\left\{ \begin{array}{l} i : X_k \rightarrow \lambda.Y\mu \quad [\alpha, \gamma] \\ Y \rightarrow \nu \end{array} \right. \Longrightarrow i : Y \rightarrow .\nu \quad [\alpha', \gamma']$$

$$\begin{array}{rcl} \alpha' & = & \sum_{\forall \lambda, \mu} \alpha(i : X_k \rightarrow \lambda.Y\mu)P(Y \rightarrow \nu) \\ \gamma' & = & P(Y \rightarrow \nu) \end{array}$$

The prediction step introduces the rule probabilities $P(Y \rightarrow \nu)$ associated with productions $Y \rightarrow \nu$ into the parsing process [3].

4.2 Extensions to uncertain terminals

If the likelihoods of the incoming symbols in the string are available, they can be incorporated into the parsing algorithm by multiplying forward and inner probabilities with

[2] Due to recursion, in some instances a recursive correction needs to be applied to probability computations. For complete details refer to [15] and [8].

[3] Recursive correction needs to be applied.

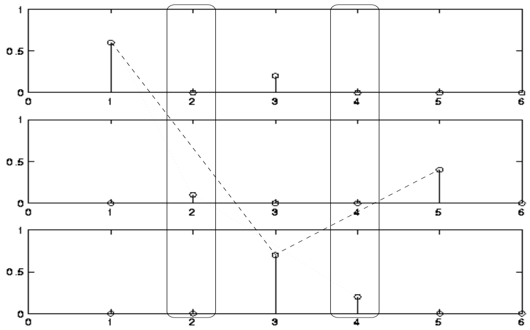

Figure 3: Example of the lattice parse input. The dashed line shows a possible parse. The two rectangles drawn around samples 2 and 4 show the "spurious symbols" for this parse and which need to be ignored for the derivation to be contiguous. We can see that if the spurious symbols are simply removed from the stream, an alternative derivation for the sequence, shown by a dotted line, will be interrupted. Sample 3 contains two concurrent symbols which are handled by the lattice parsing.

the likelihoods of the symbols at the scanning step. This essentially makes it possible to consider multiple instances of the input symbol at each time step by performing the parse on a input lattice. The input will be presented to the parser in a form of a vector consisting of the non-zero likelihoods of the input symbols at a discrete instance of time. The parsing is performed in a parallel manner for all suggested terminals. In Earley framework it can be done efficiently since with each additional terminal computational complexity only increases linearly.

4.3 Spurious Symbol Deletion.

Parsing needs to be performed on a lattice, where the symbols which we need to consider for inclusion in the string, come at random times. This results in appearance of "spurious" (i.e. ungrammatical) symbols in the input stream. We need to be able to consider these inputs separately, at different time steps and disregard them if their appearance in the input stream for some derivation is found to be ungrammatical. At the same time, we need to preserve the symbol in the stream for considering it in other possible derivations, perhaps even of a completely different string. The problem is illustrated by figure 3.

One simple approach is to convert the grammar G to a robust grammar $\hat{G}$ by first introducing the $SKIP$ non-terminal which is allowed to match any input symbol [4], and then by making the following modifications:

1. Each terminal in productions of G is replaced by a pre-terminal in $\hat{G}$:

$$\begin{array}{lcl} G: & \Rightarrow & \hat{G}: \\ A \rightarrow bC & & A \rightarrow \hat{B}C \end{array}$$

2. For each pre-terminal of $\hat{G}$ a skip rule is formed:

$$\begin{array}{ll} \hat{G}: & \\ \hat{B} & \rightarrow \quad b \quad | \quad SKIP \quad b \quad SKIP \end{array}$$

[4] including ϵ - an empty terminal

This process can be performed automatically as a preprocessing step, when the grammar is read in by the parser, so that no modifications to the grammar are explicitly written.

4.4 Constraining Terminal Length Consistency.

If we only consider the event of the terminal occurrence, without any regard to the detected length of the terminal, we might find grammatical sentences among "overlapping" terminals. In order to avoid this we need to implement an algorithm which ensures the timing consistency of the input symbols. In Earley framework, we can implement it in incremental fashion as a filter to the completion step, while keeping track of the terminal lengths during scanning and prediction.

In order to accomplish the task we need to introduce two state variables - h for "high mark" and l for "low mark". Each of the parsing steps has to be modified as follows:

Scanning

The scanning step reads the appropriate values which are available with the incoming terminal - beginning end ending samples of the terminal. Updates of l and h are propagated during the scanning as follows:

$$i : X_k \rightarrow \lambda.a\mu \quad [l,h] \implies i+1 : X_k \rightarrow \lambda a.\mu \quad [l,t]$$

where t is the index of the current sample. In addition to this, we set the time stamp of the whole new $i+1$th state set to t: $S_t = t$, to be used later by prediction step.

Completion

Similarly to scanning, the completion step advances the high mark of a state to that of the completing state, thereby extending the range of the completed non-terminal.

$$\left\{ \begin{array}{ll} j : X_k \rightarrow \lambda.Y\mu & [l,h] \\ i : Y_j \rightarrow \nu. & [l',h'] \end{array} \right. \implies i : X_k \rightarrow \lambda Y.\mu \quad [l,h']$$

This completion is performed for all states $i : Y_j \rightarrow \nu.$ subject to the filtering constraints $l' \geq h$ and $Y, X \neq SKIP$.

Prediction

Prediction step is responsible for updating the low mark of the state to reflect the timing of the input stream.

$$\left\{ \begin{array}{l} i : X_k \rightarrow \lambda.Y\mu \\ Y \rightarrow \nu \end{array} \right. \implies i : Y \rightarrow .\nu \quad [S_t, S_t]$$

Here, S_t is the time stamp of the state set, updated by the scanning step.

The essence of this filtering is that only the paths that are found to be consistent in the timing of their terminal symbols are considered. Exception is made for the skip states. They are designed to "remove" the symbols from the stream which are either temporally, or structurally inconsistent, and, therefore, are allowed to be temporally inconsistent themselves. This does not interrupt the parses of the sequence since the subsequences which form the parse remain connected via the skip states.

5 Experimental results

The parser in our system is augmented by an annotation module which lets us associate a production rule with an annotation. Each time the production is used to expand a non-terminal which has annotation attached to it, this annotation is emitted and the rule's resulting starting and ending sample indexes are optionally used for segmenting the input signal into semantically significant blocks. In this paper we present results of semantic recognition and semantic segmentation of some gestures.

5.1 Recognition and disambiguation

As an example of a simple gesture recognition and semantic disambiguation, we show a recognition of a short hand gesture which can take several possible forms. We define a "SQUARE" gesture as either left-handed (counterclockwise) or a right-handed (clockwise) gesture which consists of four parts - "TOP", "BOTTOM", "LEFT SIDE" and "RIGHT SIDE". In this formulation "TOP" and "BOTTOM", for example, are ambiguous because both of them can be formed by the same gesture. We note, however, that it can never happen in the same context. That is, if it is a right-handed square, "TOP" is a left-to-right movement and "BOTTOM" is a right-to-left one. In case of the left-handed square the definitions are reversed. We attempt to semantically disambiguate these definitions and recognize a "SQUARE" regardless of the fact that it can be either the right-handed or a left-handed square.

To describe this structure we use a simple grammar G_{square} : which reflects the ambiguity of the terminal meaning, with "skip" rules omitted for simplicity:

G_{square} :

SQUARE	$\rightarrow$	RH	[0.5]
	$\vert$	LH	[0.5]
RH	$\rightarrow$	TOP UD BOT DU	[1.0]
LH	$\rightarrow$	BOT DU TOP UD	[1.0]
TOP	$\rightarrow$	LR	[0.5]
	$\vert$	RL	[0.5]
BOT	$\rightarrow$	RL	[0.5]
	$\vert$	LR	[0.5]
LR	$\rightarrow$	left-right	[1.0]
UD	$\rightarrow$	up-down	[1.0]
RL	$\rightarrow$	right-left	[1.0]
DU	$\rightarrow$	down-up	[1.0]

We receive input data from a "Stive" vision system [17], shown in figure 5. The system uses stereo algorithms to determine x-y-z position of person's hands and head. At a frame rate of approximately 20 frames a second, the "Square" data set consists of 150 - 200 samples for each experiment.

For terminal recognition we trained four three-state HMMs on x and y velocities of 20 examples of each of the primitive hand movements. After achieving reasonable recognition rate, we performed several "SQUARE" gestures determining candidate events as described above. The results were passed to the parser yielding the results, presented in figure 4. The figure 4a shows that the semantic structure recovered was that of a right-hand square and the whole sequence was labeled as a "SQUARE". Recognition results for a left-handed square sequence are shown in figure 4b. Note that the left-right gesture was interpreted as "TOP" in the global context in the first case, and as "BOTTOM" in the second. The figures show how timing constraints propagated through the parse and formed continuous coverage of the input signal.

```
a) Segmentation <rsquare.dat>:
Label               Segment  LogP
---------------------------------------
Right hand square [0  146] 8.619816e-01
        Top       [0  23 ] 8.557740e-01
        Up down   [23 66 ] 7.490584e-01
        Bottom    [66 94 ] 8.863638e-01
        Down up   [94 146] 9.567336e-01
Viterbi probability =  0.02400375

b) Segmentation <lsquare.dat>:
Label               Segment   LogP
---------------------------------------
Left hand square  [0    173] 8.304875e-01
        Bottom    [0    49 ] 9.351195e-01
        Down up   [49   71 ] 6.933256e-01
        Top       [71   132] 7.313213e-01
        Up down   [132 173] 9.593422e-01
Viterbi probability =  0.01651770
```

Figure 4: "Square" sequence segmentation.
a) right-handed square, b) left-handed square.

5.2 Recognition and semantic segmentation

As a more complex test of our approach we chose the domain of musical conducting. It is easy to think of conducting gestures as of complex gestures consisting of a combination of simpler parts, for which we can write down a grammar (coincidentally, a book describing baton techniques, written by a famous conductor Max Rudolf [11] is called "The Grammar of Conducting"). We capture the data from a person, who is a trained conductor and who uses natural conducting gestures. The task we are attempting to solve is the following. A piece of music by Sibelius [5] includes a part with complex 6/4 music beat pattern. Instead of using this complex conducting pattern, conductors often use 2/4 or 3/4 gestures by merging 6/4 beats into groups of three or two at will. For the experiment we collect the data from a conductor conducting a few bars of the score arbitrarily choosing 2/4 or 3/4 gestures, and attempt to find bar segmentation, simultaneously identifying the beat pattern used. Experimental setup is essentially the same as in the previous example. We trained five HMMs on two primitive components of a 2/4 pattern and the three components of a 3/4 pattern. Some of the primitives in the set are very similar to each other and, therefore, corresponding HMMs show high likelihood at about the same time, which results in a very "noisy" lattice (figure 2a). We parse the lattice with the grammar G_c (again, for simplicity omitting the SKIP productions):

[5] Jean Sibelius (1865-1957), Second Symphony, Opus 43, in D Major

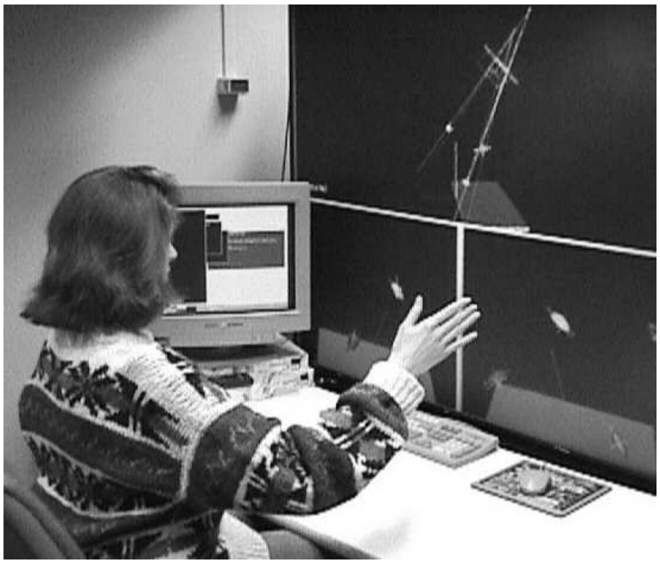

Figure 5: The Stereo Interactive Virtual Environment (STIVE) computer vision system used to collect data.

```
Segmentation:
BAR:
    2/4     start/end sample: [0 66]
    Conducted as two quarter beat pattern.
BAR:
    2/4     start/end sample: [66 131]
    Conducted as two quarter beat pattern.
BAR:
    3/4     start/end sample: [131 194]
    Conducted as three quarter beat pattern.
BAR:
    2/4     start/end sample: [194 246]
    Conducted as two quarter beat pattern.
Viterbi probability =  0.00423416
```

Figure 6: Results of the segmentation of a long conducting gesture for the bar sequence 2/4-2/4-3/4-2/4.

$$
\begin{array}{llll}
G_c: & & & \\
\text{PIECE} & \rightarrow & \text{BAR PIECE} & [0.5] \\
 & | & \text{BAR} & [0.5] \\
\text{BAR} & \rightarrow & \text{TWO} & [0.5] \\
 & | & \text{THREE} & [0.5] \\
\text{THREE} & \rightarrow & \text{down3 right3 up3} & [1.0] \\
\text{TWO} & \rightarrow & \text{down2 up2} & [1.0]
\end{array}
$$

The results of run of a lower level part of the algorithm on a conducting sequence 2/4-2/4-3/4-2/4 are shown in figure 2 with the top plot of 2a dysplayng a y positional component). Figure 6 demonstrates output of probabilistic parsing algorithm in form of semantic labels and corresponding sample index ranges.

From these figures we can see that the segmentation and labeling are performed correctly, showing a great deal of semantic filtering, where SKIP states account for large portion of the input samples.

6 Conclusions and future work

The use of formal grammars in syntactic pattern recognition is reasonable if decomposition of the entity under consideration into a set of primitives that lend themselves to automatic recognition is possible. In this paper we presented a method of combining the probabilistic model of simple "vocabulary" gestures and the higher level structural knowledge for the tasks of recognition of temporal behaviors and activities. The method is used for semantic segmentation, disambiguation and labeling complex behaviors that feature the decompositional properties.

With all the advantages of the method there are still problems that were not addressed in our work, for instance, grammar inference is hard to do in the framework of stochastic parsing. We have not researched the opportunities of fully utilizing the production probabilities. In the experiments above we determined them heuristically using simple reasoning based on our understanding of the process. The rule probabilities, which reflect "typicality" of a string derivation, will play a significantly more important role in recognizing and interpretation of activities of the higher complexity than those presented in the paper. We plan exploring the added advantages of the learned probabilities in our further research.

References

[1] A. F. Bobick and A. D. Wilson. A state-based technique for the summarization and recognition of gesture. *Proc. Int. Conf. Comp. Vis.*, 1995.

[2] Aaron F. Bobick. Movement, activity, and action: The role of knowledge in the perception of motion. In *Philosophical Transactions Royal Society London B*, 1997.

[3] M. Brand. Understanding manipulation in video. In *AFGR96*, pages 94–99, 1996.

[4] J.D. Courtney. Automatic video indexing via object motion analysis. *PR*, 30(4):607–625, April 1997.

[5] T.J. Darrell and A.P. Pentland. Space-time gestures. *Proc. Comp. Vis. and Pattern Rec.*, pages 335–340, 1993.

[6] Charniak. E. *Statistical Language Learning.* MIT Press, Cambridge, Massachusetts and London, England, 1993.

[7] K.S. Fu. A step towards unification of syntactic and statistical pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(3):398–404, May 1986.

[8] Yuri Ivanov and Aaron Bobick. Probabilistic parsing in action recognition. Tr 450, MIT Media Lab, Vision and Modeling Group, 1997.

[9] F. Jelinek, J.D. Lafferty, and R.L. Mercer. Basic methods of probabilistic context free grammars. In Pietro Laface and Renato Di Mori, editors, *Speech Recognition and Understanding. Recent Advances, Trends, and Applications*, volume F75 of *NATO Advanced Study Institute*, pages 345–360. Springer Verlag, Berlin, July 1992.

[10] L. R. Rabiner and B. H. Juang. *Fundamentals of speech recognition.* Prentice Hall, Englewood Cliffs, 1993.

[11] Max Rudolf. *The Grammar of Conducting. A Comprehensive Guide to Baton Techniques and Interpretation.* Schimmer Books, New York, 1994.

[12] Robert Schalkoff. *Pattern Recognition: Statistical, Structural and Neural Approaches.* Wiley, New York, 1992.

[13] J. Schlenzig, E. Hunter, and R. Jain. Recursive identification of gesture inputs using hidden markov models. *Proc. Second Annual Conference on Applications of Computer Vision*, pages 187–194, December 1994.

[14] T. E. Starner and A. Pentland. Visual recognition of American Sign Language using hidden markov models. In *Proc. of the Intl. Workshop on Automatic Face- and Gesture-Recognition*, Zurich, 1995.

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

[16] W. G. Tsai and K. S. Fu. Attributed grammars - a tool for combining syntatctic and statistical approaches to pattern recognition. In *IEEE Transactions on Systems, Man and Cybernetics*, volume SMC-10, number 12, pages 873–885, 1980.

[17] Christopher Wren, Ali Azarbayejani, Trevor Darrell, and Alex Pentland. Pfinder: Real-time tracking of the human body. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(7):780–785, July 1997.