Complex Activity Representation and Recognition by Extended Stochastic Grammar

Zhang Zhang, Kaiqi Huang, and Tieniu Tan

National Laboratory of Pattern Recognition, Institute of Automation,
Chinese Academy of Sciences, Beijing 100080, China
{zzhang, kqhuang, tnt}@nlpr.ia.ac.cn

Abstract. Stochastic grammar has been used in many video analysis and event recognition applications as an efficient model to represent large-scale video activity. However, in previous works, due to the limitation on representing parallel temporal relations, traditional stochastic grammar cannot be used to model complex multi-agent activity including parallel temporal relations between sub-activities (such as "during" relation). In this paper, we extend the traditional grammar by introducing Temporal Relation Events (TRE) to solve the problem. The corresponding grammar parser appending complex temporal inference is also proposed. A system that can recognize two hands' cooperative action in a "telephone calling" activity is built to demonstrate the effectiveness of our methods. In the experiment, a simple method to model the explicit state duration probability distribution in HMM detector is also proposed for accurate primitive events detection.

1 Introduction

Activity recognition in video is a key problem in many computer vision applications, such as video indexing and retrieval, intelligent surveillance, human computer interaction and intelligent robot. In most previous works, the bottom-up method based on statistical learning is widely used. Some statistical tools were used to model the state transition process in the feature space. HMM and DBN are the typical statistical models. In early work [5], HMM was used to recognize the sign language. In [6], the authors exploited Coupled Hidden Markov Models to model more complicated interaction activity. In [7], the authors developed a Dynamically Multi-Linked Hidden Markov Model (DML-HMM) to interpret group activities involving multiple objects captured in an airport scene.

These statistical methods can model the activity automatically, but most works focus on the short-term activities, and the basic premise of the approach is that the observed feature sequence can be considered Markovian. However, the pure bottom-up method may fail due to the need of a large training data and the huge computation complexity of the model structure, when we consider a large temporal scale activity.

Fortunately a complex large-scale activity often can be considered as a combination of several simple sub-activities that have explicit semantic meanings. So, as proposed in [3], the activity recognition task can be split into two steps: first bottom-up statistical method can be used to detect simple sub-activities. Then the prior structure knowledge is used to construct a composite activity model.

In some examples, the prior structure knowledge has been combined into activity model successfully. In [9], the authors made up several concept hierarchies of actions to describe activities in an image sequence. In [10], the authors extended CASE that was used for language understanding previously to represent complex temporal relations between sub-events. In [11], the author propose an event ontology for representing complex spatio-temporal events, then a new model called as Event Recognition Language (ERL) was defined to represent the events of interest.

Stochastic grammar is also used to embed the prior structure knowledge. In [3], the authors gave a set of special solutions for handling uncertain input and errors in primitive detection. And some composite activities in gesture and surveillance applications were used to demonstrate the effectiveness of grammar parsing. In [1], the rules of Black-jack card game were specified by stochastic context-free grammar (SCFG), which was a multi-tasked activity including two player's interaction. In [2], the Towers of Hanoi task was analyzed by stochastic grammar, and the experiments demonstrated the high-level parser could give feedback to influence low level tracking result. In these studies, grammar method has shown some advantages for activity representation: (1) Event structures can be simply embedded in grammar productions. The representation is concise and easy understanding; (2) Many efficient parsing algorithms have been studied in speech recognition and natural language processing, which can be used for reference in computer vision; (3) Previous works have given good solutions to some problems appearing in computer vision application, such as the uncertainty input and various detection errors.

However, traditional stochastic grammar is only suited for the specification of temporally well-ordered activities. But different agents may play different roles simultaneously in many interaction activities. Traditional stochastic grammar may fail to encode these parallel relations in the interaction activities.

In order to solve the above problem, we extend traditional stochastic grammar by introducing Temporal Relation Event (TRE) and designing the corresponding parsing algorithm. Experiments have demonstrated the effectiveness of our methods. The main contributions of this paper are as follows:

(1) Extend traditional stochastic grammar to represent complex temporal relations and an event ontology is used to instruct the foundation of activity grammar.
(2) Modify the parsing algorithm to adapt for the extended grammar, and the new parser is compatible with traditional parsing.
(3) For primitive event detection, a simple alternative method is used to model the explicit state duration probability distribution in the HMM detector. The detector can identify the time interval of primitive events accurately.

The remainder of this paper is arranged as follows. In Section 2, the extended stochastic grammar is defined and the corresponding parsing algorithm is proposed. Section 3 introduces our experiment system architecture and the techniques to detect primitive events. Experimental results and some analysis are shown in Section 4.

2 Extended Stochastic Grammar Representation and Parsing

2.1 Grammar Representation

A hierarchical event ontology similar to that in [11] is introduced to instruct grammar's foundation. The events can be classified into three categories: primitive event, single-agent event and multi-agent event. Primitive event is the simplest event that is regarded as terminal in the grammar. Single-agent event is a combination of several primitive events with temporal sequencing. Multi-agent event is composed of single-agent events, and the composition operator may be complex temporal relation.

Different from previous works, single-agent event and multi-agent event are represented by different strategies. Traditional production is used to define single-agent event. For multi-agent event, we introduce seven temporal relation events (TRE) to handle the complex temporal relations. TREs include: "*before*", "*overlap*", "*during*", "*finish*", "*meet*", "*equal*", "*start*". The meanings can refer to Allen's interval logic relations [4]. A TRE connects two common events that belong to different agents or different agent groups involved in an interaction activity. For example, a production:

$$S \to A \quad during \quad B \quad [p] \tag{1}$$

That means if an agent completes event A, another agent completes event B, and the interval of event A is *during* the interval of event B, event S occurs. P is the conditional probability of the production being chosen.

By introducing TREs, a multi-agent event can be represented conveniently. But TREs are not actual primitive events. They are generated in the parsing process. So the parsing algorithm must be changed.

2.2 Grammar Parsing

Our parsing algorithm is derived from the Earley-Stolcke algorithm [12] and its subsequent application to computer vision by Ivanov and Bobick [3]. Some parsing details, such as uncertainty handling may be found in their works. In the following, we mainly explain our modification on the original parsing algorithm.

2.2.1 Parameters

Three parameters *agent, start* and *end* are augmented to characterize primitive event and parsing state. So a parsing state can be represented as follows:

$$i : X_j \to Y \cdot during \ Z \ [\alpha, \gamma, agent, start, end] \tag{2}$$

where *agent* indicates the executor of a primitive event or parsing state. [*start, end*] denotes the time interval of a primitive event or parsing state. α is called as forward probability, γ is called as inner probability, the dot is the marker of the current position in the input string. For simplicity, α γ's computation can refer to [3].

2.2.2 Parsing algorithm

The Earley-Stolcke algorithm [12] analyzes a symbol string iteratively through three steps: *scanning*, *completion* and *prediction*. We prefer to embed the modification in such framework to assure the compatibility with the traditional parsing.

Scanning:
According to the *agent* of the current scanned primitive event, new scanned states are generated for the current state set as follows:

$$\left\{\begin{array}{c} a\left[agent_a, start_a, end_a\right] \\ i-1: X_k \rightarrow \lambda \cdot a\mu\left[agent_s, start, end\right] \end{array}\right. \quad if\ agent_a = agent_s$$
$$\Rightarrow \left\{\begin{array}{ll} i: X_k \rightarrow \lambda a \cdot \mu\left[agent_a, start_a, end_a\right] & if\ \lambda = \varepsilon \\ i: X_k \rightarrow \lambda a \cdot \mu\left[agent_a, start, end_a\right] & otherwise \end{array}\right. \tag{3}$$

If the current scanned symbol is a TRE, the new scanned states should also be added into the *k*th state set, which is prepared for the backtracking in *completion* step.

If no state is matched successfully with the *agent* constraint, there may be a new agent appearing in the scene. Do scanning process again in those states whose *agent* is zero. Here *agent* is zero means the state has not been specified by any agents.

$$\left\{\begin{array}{l} a\ \left[agent_a, start_a, end_a\right]\ if\ agent_a\ is\ a\ new\ agent \\ i-1: X_k \rightarrow \lambda \cdot a\mu\left[0,0,0\right]\ appearing\ in\ the\ scene \end{array}\right.$$
$$\Rightarrow i: X_k \rightarrow \lambda a \cdot \mu\left[agent_a, start_a, end_a\right] \tag{4}$$

Besides the above operations, another function in scanning is to judge whether a TRE should be generated in the next *completion* step. A sign *flag* will be evaluated as *true*, if and only if the following conditions are satisfied:

a) In the last predicted state set, there is a predicted state that denotes the next scanned event should be a TRE. We define the primitive event on the right part of the TRE as the *post-event*, the left one as the *pre-event*;

b) The current scanned primitive event's *agent* is different from the state that satisfies condition a.

Completion:
If the current scanned symbol is a TRE, there will be no *completion* step, because there is no completed state in the current state set.

Otherwise, the completion process is implemented appending the *agent* constraint:

$$\left\{\begin{array}{c} i: Y_j \rightarrow v \cdot\left[agent_f, start_f, end_f\right] \\ j: X_k \rightarrow \lambda \cdot Y\mu\left[agent_m, start_m, end_m\right] \end{array}\right. \quad if\ agent_f = agent_m$$
$$\Rightarrow i: X_k \rightarrow \lambda Y \cdot \mu\left[agent_m, start_m, end_f\right] \tag{5}$$

Then, if the sign *flag* is *true*, a TRE can be generated according to the following steps:

First, in the current state set, some states whose production head is identical with the *post-event* are selected out for the next temporal inference.

Then for each state selected in the last step, if the marker has been at the rightmost position of the production, which means *post-event* has been completed, two values are computed to measure the temporal relation between *pre-event*'s interval and *post-event*'s interval.

$$\eta_1 = \left(post.start - pre.start\right) / \left(pre.end - pre.start\right) \tag{6}$$

$$\eta_2 = (post.end - pre.end) / (pre.end - pre.start) \tag{7}$$

where the *post-start* means *post-event*'s start point, *post.end* means the end point; *post-start* means *pre-event*'s start point, *post.end* means the end point.

Finally, a TRE is generated according to Table 1. The threshold is selected empirically. If the *post-event* has not been completed, the end time must be later than the *pre-event*'s end time, which can be equal to $\eta_2 > 0.1$ So TRE also can be generated according to Table 1. The TRE's *agent* is to be specified by a group agent: one is the *pre-event*'s *agent*; another is the *post-event*'s *agent*.

Table 1. TRE generation

Conditions	TRE
$\eta_2 > 0.1 \wedge \eta_1 > 1.1$	*before*
$\eta_2 > 0.1 \wedge 0.9 < \eta_1 \leq 1.1$	*meet*
$\eta_2 > 0.1 \wedge 0.1 < \eta_1 \leq 0.9$	*overlap*
$\eta_2 > 0.1 \wedge -0.1 < \eta_1 \leq 0.1$	*start*
$\eta_2 > 0.1 \wedge \eta_1 \leq -0.1$	*during*
$\eta_2 \leq 0.1 \wedge \eta_1 > 0.1$	*i-finish*
$\eta_2 \leq 0.1 \wedge \eta_1 \leq -0.1$	*finish*
$\eta_2 \leq 0.1 \wedge -0.1 < \eta_1 \leq 0.1$	*equal*

In the next iteration, the TREs become the scanned events. The event that triggered the generation of TRE should be handled again after the scanning of TRE.

Prediction:

If the current scanned primitive event is a TRE, the prediction step only rewrites all the predicted states in the last state set into the current state set. Otherwise, common prediction is processed.

If the predicted production includes a TRE, the *post-event* of the TRE should also be predicted:

$$\begin{cases} i: X_k \to \lambda \cdot Z\mu\,[agent, start, end] \\ Y \to v\omega N \qquad \text{if } \omega \text{ is a TRE} \\ M \to \psi \end{cases}$$
$$\Rightarrow \begin{cases} i: Y_i \to \cdot v\omega N\,[agent, 0, 0] \\ i: M_i \to \cdot\psi\,[0,0,0] \end{cases} \tag{8}$$

where $R_L(Z \Rightarrow {}_L Y)$ and $R_L(N \Rightarrow {}_L M)$ are both nonzero. Here, the R_L is the left corner relation matrix [12].

Finally, all predicted states in the last state set whose *agent* is different from the current scanned primitive event should be rewritten in the current state set, which are prepared for handling other agent's event in the next parsing iteration.

3 System Architecture and Primitive Event Detection

3.1 System Architecture

The system architecture is shown in Figure 1. First, the video signal is fed into the object detection and tracking module (red part of Figure 1) that includes the whole low level processing. In this module, we can robustly obtain the moving object's position, direction and other low level features. Then each moving object's feature sequence (such as trajectory, etc) is fed into the primitive event detection module (green part of Figure 1). In here, we train a HMM for each primitive event and a backward-looking algorithm is implemented to detect whether a primitive event has occurred. Once a primitive event has been detected, it is sent to the grammar parser module (blue part of Figure 1) where the event is analyzed with context information. Finally, higher semantic interpretation is inputted.

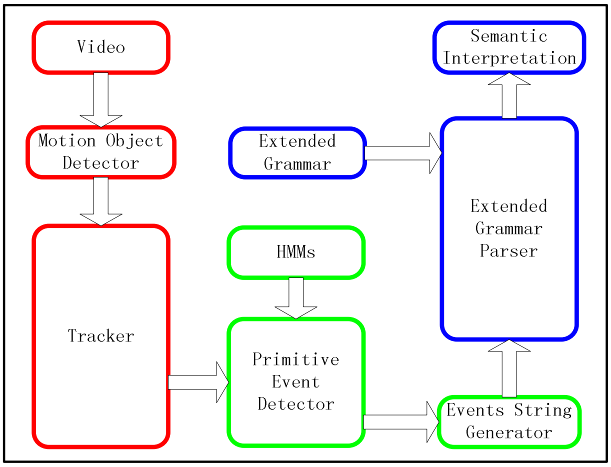


Fig. 1. System architecture

In our experiment, a tracking system was constructed, which used mean-shift tracker to get the robust multiple objects' trajectories and other low-level features. The details on the tracking method may be found in [8].

3.2 Primitive Event Detection

As shown in [3], we also choose HMM as the primitive events' detector. However traditional HMM has the limitation on modeling the state duration [13], which often leads to inaccurate detection results. In our experiment, the detected errors between the detected results and the ground truths can reach nearly 30 frames if we only use pure HMM as the primitive event detector. Such errors are unacceptable for the temporal inference in grammar parsing. For this problem, we use a simple histogram method to measure the state duration density. Different with directly measuring from the training sequences used in the segmental k-means procedure [13], we count each

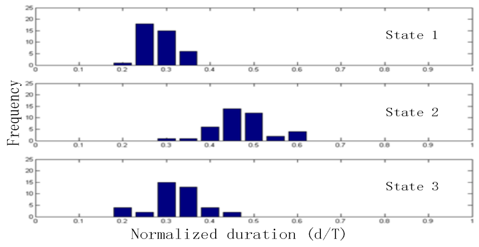


Fig. 2. Normalized duration Histograms for the 3-state HMM of a primitive event in our experiment

state's duration in Viterbi path to acquire each state's duration histogram. A histograms for a 3-state HMM of a primitive event in our experiment is shown in Figure 2.

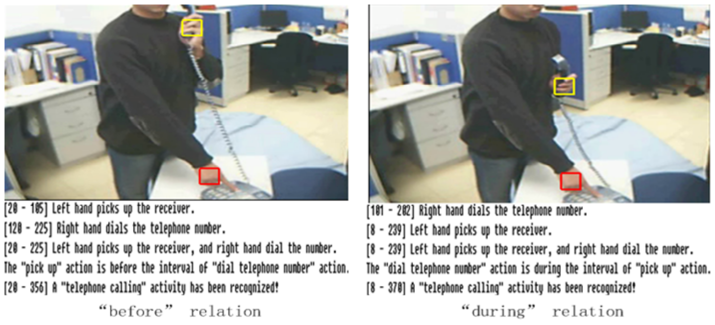
For primitive event detection, first the Viterbi path and the corresponding probability is computed for an observed sequence. Then as proposed in [13], a post-processor is added to the Viterbi probability according to the explicit state duration histogram:

$$\log \tilde{p}(q,O \mid \lambda) = \log p(q,O \mid \lambda) + \sum_{j=1}^{N} \log p_j(d_j) \tag{9}$$

where d_j is the duration of state j along the Viterbi path, $p(q,O \mid \lambda)$ is the Viterbi probability and $p_j(d_j)$ is the duration probability of state j.

4 Experimental Results and Analysis

Here, "telephone calling" activity is introduced to demonstrate the effectiveness of our methods. In our study, a telephone calling activity can be decomposed into two hands' cooperative actions. The left hand and right hand are regarded as two agents. It is our experiment's target to recognize the temporal relation between "pick up phone" and "dial telephone number". The temporal relations between the two sub-events can be described as variant types: *before, during* and *overlap.*



```
[20 - 105] Left hand picks up the receiver.
[120 - 225] Right hand dials the telephone number.
[20 - 225] Left hand picks up the receiver, and right hand dial the number.
The "pick up" action is before the interval of "dial telephone number" action.
[20 - 356] A "telephone calling" activity has been recognized!
```

```
[101 - 202] Right hand dials the telephone number.
[8 - 239] Left hand picks up the receiver.
[8 - 239] Left hand picks up the receiver, and right hand dial the number.
The "dial telephone number" action is during the interval of "pick up" action.
[8 - 370] A "telephone calling" activity has been recognized!
```

Fig. 3. Two types of "telephone calling" activities

Productions	Description
S → X b N [1.0]	Tele call → call to somebody *before* put down telephone
X → U b I [0.5]	Call to somebody → pick up phone *before* dial number
X → U o I [0.1]	Call to somebody → pick up phone *overlap* dial number
X → I g U [0.4]	Call to somebody → dial number *during* pick up phone
U → r p h t [0.5]	Pick up phone
U → r e [0.5]	Pick up phone
N → d w [1.0]	Put down phone
I → r a w [1.0]	Dial telephone number

Terminals	Description
r	One hand moves from body to phone
p	One hand moves from phone to bosom
h	One hand hangs around bosom
t	One hand moves from bosom to ear
e	One hand moves from phone to ear
d	One hand moves from ear to phone
w	One hand moves from phone to body
a	One hand linger over phone

Terminals	Description
b	“Before” relation
o	“Overlap” relation
g	“During” relation

Fig. 4. “Telephone calling” activity grammar

Two types of “telephone calling” activities are presented in Figure 3. The left picture shows the "pick up" action is *before* the "dial telephone number" action. The right one shows the "dial telephone number" action is *during* the "pick up" action. According to prior knowledge, we obtain the activity grammar, as shown in Figure 4.

We have recorded 12 consecutive “telephone calling” activities. Using our tracking system, 48 trajectories are acquired by specifying different initial tracking position. Among these 48 trajectories, 40 consecutive trajectories are used to train HMMs and explicit state duration model. Other trajectories are used to test. For each primitive event, we train 3-states HMM with a Gaussian Mixture output probability.

To compare primitive event detection accuracy between pure HMM and HMM with explicit state duration model, the results are evaluated by labeling each frame. For each frame a values is evaluated that specify whether a particular primitive event is active or inactive. By comparing these labels with ground truth, we can compute the overall-correct ratio as [*correct_positive*+ *correct_negative*]/[*all_frames*]. The statistical results are available in the left part of Table 2. As shown in this table, the detector with explicit duration model is more accurate than the pure HMM detector.

The middle part of Table 2 shows the results on primitive event detection. The right part is TREs recognition results through grammar parsing. We can find that the temporal relations are all recognized successfully, and all the true primitive events are recognized, but there are many insertion errors in primitive event detection. Two reasons may lead to the problem: (1) due to the influence of viewing angle, the trajectories of some primitive events are very similar in the image coordinate. So a trajectory segment may be received by two HMMs simultaneously. For example, the trajectory of action “w” is very similar to the trajectory of action “p” in our experiments. (2) Some primitive events are just a part of other primitive events in the trajectory space. For example, the trajectory the action “t” is just a part of the action “e”. But these errors can be corrected completely through grammar parsing, as shown in the last column of middle part of Table 2.

Table 2. Recognition results. The left and middle parts are the results for primitive events detection. The right part denotes the recognition result of TREs. The symbols' meaning can be referred to Figure 4.

Primitive events	HMM	ED-HMM
r	95. 4%	97. 4%
p	90. 4%	93. 6%
h	88. 8%	90. 6%
t	94. 7%	96. 8%
e	81. 3%	96. 8%
d	92. 5%	99. 5%
w	94. 9%	96. 4%
a	96. 1%	96. 3%
Average	91. 7%	95. 9%

Primitive events	Ground truth	Actual test	Error rate	Recovery rate
r	16	24	33%	100%
p	4	4	0%	–
h	4	4	0%	–
t	4	8	50%	100%
e	4	4	0%	–
d	8	8	0%	–
w	16	24	33%	100%
a	4	4	0%	–

Temporal relation	Ground truth	Actual test	Error rate
b	11	11	0%
o	1	1	0%
g	4	4	0%

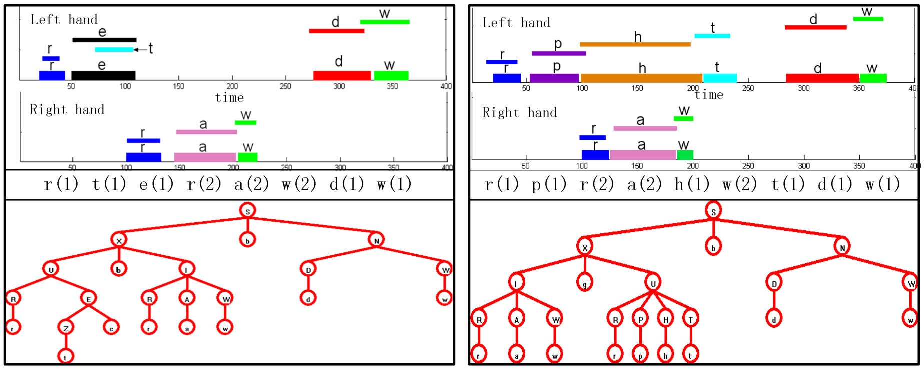

Left hand
Right hand
time
r(1) t(1) e(1) r(2) a(2) w(2) d(1) w(1)
r(1) p(1) r(2) a(2) h(1) w(2) t(1) d(1) w(1)

Fig. 5. Illustration of the "telephone calling" activity recognition process

Two activity recognition processes are shown in Figure 5. In the left part, the primitive events detection results are shown at the top, including left hand and right hand's actions. Compared with the ground truth (the thick line), the detection result is accurate enough for the temporal inference. The symbol string in the middle part is the primitive events string, the number in the bracket represent the primitive events' *agent* ("1" represents left hand, "2" two represents right hand). The events are ordered by their end times. At the bottom, the parsing tree is shown. The symbol "Z" and other capital letters of terminals ("R", etc) are all derived by the *skip* rule that is used for correcting insertion errors [3]. As we can see, the whole activity is recognized successfully, the "U" action (pick up) is *before* the "I" action (dial telephone number), and the insertion error "t" is corrected. The right part of Figure 5 shows another type of "telephone calling" activity, the temporal relation "*during*" between the "I" action (dial telephone number) and the "U" action (pick up) is also recognized successfully.

5 Conclusion

In this paper, we have extended the traditional stochastic grammar and designed the corresponding parsing algorithm to recognize complex activities involving parallel temporal relations. We have applied the extended stochastic grammar parser in an activity recognition system. For accurate primitive event detection, a simple method has been proposed to model the explicit state duration probability distribution in the HMM detector. Experimental results have demonstrated the validity of our method. An advantage of the two phases strategy is also shown: higher grammar parsing can correct the errors in lower primitive event detection that may be difficult to be identified only using image features due to some reasons (viewing angle, etc).

In this work, the grammar is defined manually. In the future work, the association rules between primitive events may be learned using some data mining techniques. The quantitative description of the temporal relations is also our future work.

Acknowledgment

This work is supported by research grants from the National Basic Research Program of China (No. 2004CB318100), the National Natural Science Foundation of China (No. 60335010) and the International Cooperation Program of Ministry of Science and Technology of China (Grant No. 2004DFA06900).

References

1. D.Moore, I.Essa, "Recognizing multitasked activities using stochastic context-free grammar", *CVPR WM vs. CV 2001*,Kauai, Hawaii, Dec 2001
2. D.Minnen, I.Essa, T.Starner, "Expectation Grammars: Leveraging High-Level Expectations for Activity Recognition", *Proc. CVPR2003*, vol.2.pp.626-632 Madison, WI, June 2003
3. Y.A.Ivanov, A.F.Bobick, "Recognition of visual activities and interactions by stochastic parsing", *IEEE TRANS.PAMI*, vol.22.no8, pp.852-872, Aug 2000
4. J.F. Allen, F. Ferguson, "Actions and events in interval temporal logic" *J. Logic and Computation* Volume 4, Number 5, pp. 531-579, Otc 1994
5. T.Starner, A.Pentland, "Real-Time American Sign Language Recognition From Video Using Hidden Markov Models" *Perceptual Computing Section Technical Report* No. 375, MIT Media Lab, Perceptual Computing Group, 1996.
6. N.M.Oliver, B.Rosario, A.P.Pentland, "A Bayesian Computer Vision System for Modeling Human Interactions" *IEEE TRANS.PAMI*, vol.22.no8, pp.831-843, Aug 2000
7. S. Gong, T. Xiang, "Recognition of group activities using dynamic probabilistic networks" *Proc. ICCV2003*, vol.2, pp. 742-749, Nice, France, Oct 2003
8. D. Comaniciu, V. Ramesh, P. Meer, "Real-time tracking of non-rigid objects using mean shift" *Proc. CVPR2000*, vol.2, pp.142-149, 2000
9. A.Kojima, T.Tamura, "Natural Language Description of Human Activities from Video Images Based on Concept Hierarchy of Actions ", *IJCV*, vol 50, num.2, PP.171-184, Nov 2002
10. A.Hakeem, Y.Sheikh, M.Shah "$CASE_E$: A Hierarchical Event Representation for the analysis of Videos" *Proc. AAAI2004*, pp.263-268, San Jose 2004
11. R.Nevatia, T.Zhao, S.Hongeng, "Hierarchical Language based Representation of Events in Video Streams" *Proc. CVPR03 Workshop on Event Mining*, Madison, Wisconsin 2003
12. A. Stolcke, "An efficient probabilistic context-free parsing algorithm that computes prefix probabilities", *Computational Linguistics*, v.21 no.2, pp.165-201, June 1995
13. L.R. Rabiner, "A tutorial on Hidden Markov Models and selected applications in speech recognition" *Proceedings of the IEEE*, Vol.77, No.2, pp.257–286, Feb 1989