Int J Comput Vis (2009) 82: 1–24
DOI 10.1007/s11263-008-0181-1

# Semantic Representation and Recognition of Continued and Recursive Human Activities

**M.S. Ryoo · J.K. Aggarwal**

Received: 10 July 2007 / Accepted: 1 October 2008 / Published online: 12 November 2008
© Springer Science+Business Media, LLC 2008

**Abstract** This paper describes a methodology for automated recognition of complex human activities. The paper proposes a general framework which reliably recognizes high-level human actions and human-human interactions. Our approach is a description-based approach, which enables a user to encode the structure of a high-level human activity as a formal representation. Recognition of human activities is done by semantically matching constructed representations with actual observations. The methodology uses a context-free grammar (CFG) based representation scheme as a formal syntax for representing composite activities. Our CFG-based representation enables us to define complex human activities based on simpler activities or movements. Our system takes advantage of both statistical recognition techniques from computer vision and knowledge representation concepts from traditional artificial intelligence. In the low-level of the system, image sequences are processed to extract poses and gestures. Based on the recognition of gestures, the high-level of the system hierarchically recognizes composite actions and interactions occurring in a sequence of image frames. The concept of hallucinations and a probabilistic semantic-level recognition algorithm is introduced to cope with imperfect lower-layers. As a result, the system recognizes human activities including 'fighting' and 'assault', which are high-level activities that previous systems had difficulties. The experimental results show that our system reliably recognizes sequences of complex human activities with a high recognition rate.

M.S. Ryoo (✉) · J.K. Aggarwal
Computer and Vision Research Center, The University of Texas at Austin, Austin, TX, USA
e-mail: mryoo@ece.utexas.edu

J.K. Aggarwal
e-mail: aggarwal@ece.utexas.edu

**Keywords** Human activity recognition · Event detection · Semantic-level video analysis · Hierarchical action recognition

## 1 Introduction

A high-level understanding of human activities is essential for various applications, including surveillance systems and human computer interaction systems. In particular, a human activity recognition system may enable the detection of abnormal activities as opposed to the normal activity of persons using public places such as airports and subway stations. The automated recognition of human activities may also be useful for real-time monitoring of elderly people, patients, and babies. In this paper, we develop a general recognition framework for high-level human activities and apply the framework to actual sequences of videos, while especially focusing on the semantic-level aspect of the recognition. Methodologies developed for representation and recognition of human actions and human-human interactions are presented and discussed throughout the paper.

The paper introduces a new probabilistic description-based approach for the recognition of high-level human activities. Our approach is to incorporate humans' conceptual knowledge of the structure of human activities into the recognition process, by enabling the system to maintain formal programming language-like representations of human activities. Our representation explicitly describes the temporal and spatial structure of human activities that the system aims to recognize using a context-free grammar (CFG) based representation scheme: a human activity is represented by decomposing it into multiple sub-events and by specifying their temporal, spatial, and logical relationships. A sub-event of one activity may be composed of multiple sub-events of itself, capturing the hierarchical structure of human activities. Once human activities are hierarchically represented, our method is able to recognize them by performing semantic matching between the representation and the observed images from a given sequence in a video. The probability (i.e. confidence) associated with an activity being matched is also computed for reliable recognition of the activity.

The process of our approach can be clearly illustrated with the following example: the recognition of the human-human interaction ‘assault’. The interaction ‘assault’ indicates a situation where one person suddenly approaches and starts attacking another. Following our description-based approach, we are able to represent the ‘assault’ in terms of three sequential sub-events, ‘approach’, ‘attacking’, and ‘fighting’, which may be decomposed themselves into their sub-events until the atomicity is obtained. As a result, the ‘assault’ corresponds to several atomic-level actions, ‘stretching a hand’ or ‘raising a hand’ for example, which are organized sequentially or concurrently. Therefore, detecting these atomic actions with statistical models (e.g. hidden Markov models) and hierarchically matching those results with the representation will enable the system to recognize the high-level activity ‘assault’. Further, based on the probability of the occurrence of atomic actions, the probability for the ‘assault’ maybe computed, measuring the confidence of the match.

The contribution of this paper is on the development of the general recognition methodology for complex high-level activities and the implementation of the framework for processing of real videos. Our approach has several advantages that distinguishes it from previous approaches. First, our approach is able to represent and recognize human activities of any level of hierarchy. Such an ability enables our approach to recognize extremely high-level activities such as ‘assault’ and ‘fighting’, which has not been attempted by previous approaches. Secondly, our approach is designed to recognize activities composed of sub-events having complex sequential and concurrent relationships. Concurrent sub-events must be represented in order to recognize high-level activities with a complex structure, while most of the previous approaches including previous statistical approaches (using dynamic Baysian networks or hidden Markov models) and syntactic approaches (using stochastic context-free grammars) were limited in the case of recognizing activities with concurrent sub-events. Finally, our approach is probabilistic, compensating for the failures of its low-level recognitions (the gesture layer may fail in the recognition of one gesture for example). Even though probabilistic frameworks have been common for statistical approaches, research to integrate a probabilistic decision into a description-based approach handling complex concurrency has not been conducted previously.

Based on our new framework and algorithms, we constructed a human activity recognition system and tested the system with real videos containing sequences of interactions between two persons. The system successfully recognized relatively simple two person interactions such as ‘pushing’ and ‘hugging’ with high accuracy. Furthermore, the experimental section of this paper also illustrates the recognition results of complex recursive activities like ‘fighting’, ‘greeting’, ‘assault’, and ‘pursuit’ that previous system have not attempted to recognize.

The paper is organized as follows. In Sect. 2, we present works done by other researchers while comparing their work to our new approach. Next, we describe the overall framework of our system, and illustrate the role of each component in Sect. 3. Section 4 illustrates the low-level components for pose and gesture estimation of persons. Sections 5 and 6 presents detailed recognition methodology used in the high-level of our system, the semantic layer. Section 5 illustrates a formal representation scheme to describe the temporal, spatial, and logical structure of high-level human activities. In Sect. 6, the hierarchical methodology to recognize represented activities from video sequences are presented. Section 7 discusses the recognition of activities with recursive and continuous characteristics (e.g. fighting). Techniques to enable the probabilistic recognition of activities are presented in Sect. 8. Experimental results are provided in Sect. 9 with real-world videos of human activities, and Sect. 10 concludes the paper.

## 2 Related Works

Various methodologies have recently been developed toward the recognition of high-level activities. Before presenting our approach, we discuss the previous approaches especially focusing on their abilities and limitations. Based on techniques that they use for recognition, approaches are classified into three categories: statistical approaches, syntactic approaches, and description-based approaches. Our approach, which belongs to the class of ‘description-based approach’, is compared not only with statistical approaches and syntactic approached, but also with other description-based approaches.

### 2.1 Statistical Approaches

The recognition of human activities using statistical models such as Bayesian networks (BNs), hidden Markov models (HMMs) and dynamic Bayesian networks (DBNs) has been studied thoroughly (Fine et al. 1998; Natarajan and Nevatia 2007; Nguyen et al. 2005; Oliver et al. 2000; Park and Aggarwal 2004a, 2004b; Shi et al. 2004; Starner and Pentland 1995; Yamato et al. 1992). In the case of statistical approaches, one statistical model is generally constructed for each activity. For each model, the probability of the model generating an observed sequence of feature vectors is calculated to measure the likelihood between the corresponding activity and a given input image sequence.

Park and Aggarwal (2004a, 2004b) presented a hierarchical framework to recognize human actions and interactions from pixel level images. The framework abstracts image sequences into poses, gestures, actions, and interactions. Their system uses extracted body part knowledge to estimate poses for each frame, and then estimates the most dominant gesture based on a sequence of poses. Bayesian networks are constructed to estimate the poses, and dynamic Bayesian networks are trained to recognize gestures. A gesture recognized through DBNs is directly converted into a single action, represented as the operation triplet. Two concurrent operation triplets of different persons are combined to form interactions, and two interactions might be combined to present the cause and effect of interactions. Their system successfully recognized atomic components of human interactions, gestures, and interactions composed of one or two gestures.

Note that our system presented in this paper takes advantage of their low-level recognition framework (Park and Aggarwal 2004a) for the gesture recognition. The pose estimation and gesture detection methodologies of our system is similar to that of Park and Aggarwal's system. The difference is in the high-level of the system to recognize human activities: Park and Aggarwal's (2004b) system used a decision tree to classify simple human interactions, while the high-level of our system uses a description-based approach that recognizes human activities by maintaining description on activities' structure. As a result, instead of recognizing interactions composed of two sequential actions as Park and Aggarwal's (2004b) system have done, our system recognizes hierarchical activities composed of multiple subevents having complex temporal, spatial, and logical structures.

Nguyen et al. (2005) used hierarchical HMMs in order to recognize two levels of actions. Similar to Park and Aggarwal's (2004a) work, they recognized primitive behaviors based on HMMs. Treating the recognition of primitive behaviors as observations for complex behaviors, they constructed another layer of HMMs on top of the primitive behavior recognition system. As a result, their system was able to represent and recognize two levels of behaviors with a probabilistic model.

Shi et al. (2004) tried to overcome the disadvantage of previous systems through using propagation networks (which can be interpreted as an extension of HMMs) as a representation of actions. Their work also decomposes actions into several atomic actions, and constructs a network describing the temporal order needed among them. Unlike HMMs, their network allowed multiple activations of nodes, implying that they are able to deal with sub-events which are occurring simultaneously.

Statistical approaches are especially suitable when recognizing sequential activities. With enough training data, statistical models are able to reliably recognize corresponding activities even in the case of noisy inputs. The major limitation of statistical approaches are their inherent inability to recognize hierarchical activities with complex temporal structures, such as an activity composed of concurrent sub-events. For example, HMMs and DBNs have difficulty modeling the relationship of activity A occurred 'during', 'started with', or 'finished with' activity B. The edges of HMMs or DBNs specify the sequential order between two nodes, suggesting that they are suitable for modeling sequential relationships, not concurrent relationships. In addition, as an activity gets more complex, statistical approaches need a greater amount of training data, preventing the approach from being applied to highly complex activities.

## 2.2 Syntactic Approaches

Syntactic approaches model human activities as multiple production rules generating a string of symbols, and adopt parsing techniques from the field of programming language to recognize the activities from a given string. Context-free grammar (CFG) and stochastic context-free grammar (SCFG) have been adopted by previous researchers to recognize high-level activities (Bobick and Wilson 1997; Ivanov and Bobick 2000; Joo and Chellappa 2006; Minnen et al. 2003; Moore and Essa 2002). Production rules of CFGs naturally lead to a hierarchical recognition of the activities, where each symbol of the string is considered as a simple action.

Ivanov and Bobick (2000) proposed a hierarchical approach for the recognition of high-level activities, which models human activities with a stochastic context-free grammar (SCFG). They divided the framework into two layers: the lower layer using HMMs for the recognition of simple actions, and the higher layer using stochastic parsing techniques for the recognition of high-level activities. The recognition result of the lower layer of the system is converted as a sequence of simple actions. The higher layer treats the sequence as a string of simple actions, enabling the parsing techniques to be applied. The overall recognition process is done probabilistically, since the activities are represented in terms of stochastic production rules. Minnen et al. (2003) also adopted SCFG for the recognition. Their system focuses on the segmentation problem of multiple objects. They have shown that the semantic-level processing of activities using CFG may help the segmentation and the tracking of objects. The concept of hallucinations are introduced to explicitly compensate for the failures of atomic-level recognition failures. Moore and Essa (2002) used CFG for the recognition of activities as well, focusing on multi-task activities.

The main limitation of syntactic approaches is also in the recognition of concurrent activities. Syntactic approaches are able to probabilistically recognize hierarchical activities composed of sequential sub-events, but are inherently limited on activities composed of concurrent sub-events. Since syntactic approaches are modeling a high-level activity as a string of atomic-level activities composing them, temporal ordering of atomic-level activities has to be strictly sequential. In addition, for their system, the user must provide all possible production rules for all possible events, even for large domains.

## 2.3 Description-Based Approaches

Description-based approaches are approaches that recognize human activities by maintaining their description (or representation) on the temporal and spatial structure of the activities which they want to recognize (Hongeng et al. 2004; Nevatia et al. 2004; Pinhanez 1999; Siskind 2001; Vu et al. 2003). They represent a high-level human activity in terms of relationships between simpler activities (i.e. sub-events) composing the activity. In description-based approaches, a time interval is usually associated with an occurring sub-event to specify necessary temporal relationships among sub-events. That is, description-based approaches model a human activity as an occurrence of its sub-event (which might be composed of their own sub-events) that satisfies certain temporal and spatial relationships.

Allen's temporal predicates (Allen and Ferguson 1994) have been widely adopted for these approaches to specify relationships (sequential, concurrent, and their combinations) between time intervals explicitly. These relationships not only include simple sequential relations between two time intervals, but also combinations of complex concurrent relations. In description-based approaches, CFG is widely used as a formal syntax of the representation (similar to those of programming languages), and an approximation algorithm for the constraint satisfaction problem is generally designed for the recognition of represented activities. Even though description-based approaches often take advantage of CFG as well, their usage and syntactic approaches' usage of CFG is completely different. Syntactic approaches use CFG for the recognition directly. Description-based approaches use CFG as a formal syntax of the representation (note that the syntax does not have to be expressed in terms of CFG).

Siskind (2001) adopted Allen's temporal predicates and constructed his event logic for the computer vision problem, recognizing block stacking events. He extracted force-dynamic relations between participants as features of events, and applied event logic directly on top of them for representation and recognition. His event logic was particularly designed for processing liquid and semi-liquid events, which our system also recognizes in Sect. 8.3. He focused on developing an efficient algorithm to recognize events that have characteristics of liquidity. His system is deterministic, recognizing high-level activities assuming that the low-level recognitions are done correctly. The major disadvantage of his system is that it limits one time interval to be used no more than once when describing temporal relationships, reducing the expressiveness of their representation.

Hongeng et al. (2004) constructed a representation language for general events, using Allen's temporal predicates, spatial predicates, and logical predicates. Their representation provided promising results on recognition of composite events, dividing the hierarchy of events into three levels. They not only provided a representation scheme, but also illustrated the initial results of the recognition system using their representation. The recognition of an activity is performed probabilistically by assuming conditional independence among its sub-events (i.e. they computed the probability of an activity as a 'product' of those of its sub-events). The major limitation of their representation language is that their hierarchy of events is strictly fixed to three levels. This limits constructing high-level composite actions from simpler composite actions, and high-level interactions from simpler interactions. In addition, their system is not able to compensate for the failure of recognition of one of its sub-events because of their conditional independence assumption.

Vu et al.'s (2003) approach was also description-based. Similar to Hongeng et al. (2004), they represented activities (which they call scenario representation) by specifying necessary conditions using Allen's temporal predicates (Allen and Ferguson 1994) and other spatial predicates. Notably, their representation is able to describe high-level activities with any levels of hierarchy. They have shown successful experimental results with the activity of a person stealing from a bank. However, unlike Hongeng et al.'s (2004) system, only conjunctive predicates are allowed when concatenating multiple temporal relationships (i.e. only 'and' concatenations allowed, not 'or'). Also, similar to the work done by Siskind (2001), their method does not define time intervals of the activity being represented explicitly, and uses a time interval that covers time intervals of entire sub-events as a resulting time interval. This prevents the system from naturally representing activities with cause-and-effect relationships (since the effects usually occurs 'after' the activity) and activities with a recursive structure. Their system is deterministic.

Table 1 compares the abilities of previous approaches and our new methodology. As shown in the table, the system we design and implement in this paper is able to recognize high-level human activities with any levels of hierarchy. The range of activities that our system is able to represent is broader as compared to the previous approaches, enabling

**Table 1** A table comparing abilities of previous systems and our new approach

| Approaches | Levels of hierarchy | Complex temporal relations | Complex logical concatenations | Recognition of recursive activities | Handle imperfect low-levels |
|---|---|---|---|---|---|
| Statistical (Nguyen et al. 2005; Park and Aggarwal 2004a; Shi et al. 2004) | Limited (depends on data amount) | | | | v |
| Syntactic (Ivanov and Bobick 2000; Minnen et al. 2003; Moore and Essa 2002) | Unlimited | | | v | v |
| Siskind (2001) | Unlimited | A sub-event participates only once | v | | |
| Hongeng et al. (2004) | Limited (3-levels) | v | v | | |
| Vu et al. (2003) | Unlimited | v | Conjunctions only | | |
| Our approach | Unlimited | v | v | v | v |

the recognition of high-level activities composed of sub-events with logical concatenations of complex temporal relationships (especially concurrent relationships). Recursive activities are also represented and recognized as a consequence. Furthermore, our proposed system has the ability to handle noisy recognition results from the low-level, gesture recognition for example. The recognition system which has all of above mentioned abilities had not been developed previously.

## 3 Framework

Overall, the system can be divided into two levels: the low-level corresponding to three layers (the body-part layer, the pose layer, and the gesture layer), and the high-level corresponding to the semantic layer. The low-level of the system recognizes atomic components of human activities, i.e. gestures, based on the sequence of input frames using various existing computer vision techniques. The high-level of the system, the semantic layer, is designed to maintain the semantic structure (temporal, spatial, and logical) of human activities encoded by a human expert. In the semantic layer, human activities are finally recognized by hierarchically matching recognition results given from the low-level with the representation of activities defined by the user, while considering the probability of matching. Figure 1 shows the overall framework of our system. Our focus in this paper is on the semantic layer.

Any technique can be used for the low-level of the system, if it correctly detects a starting point and an ending time of an occurring gesture. In order for the semantic layer to

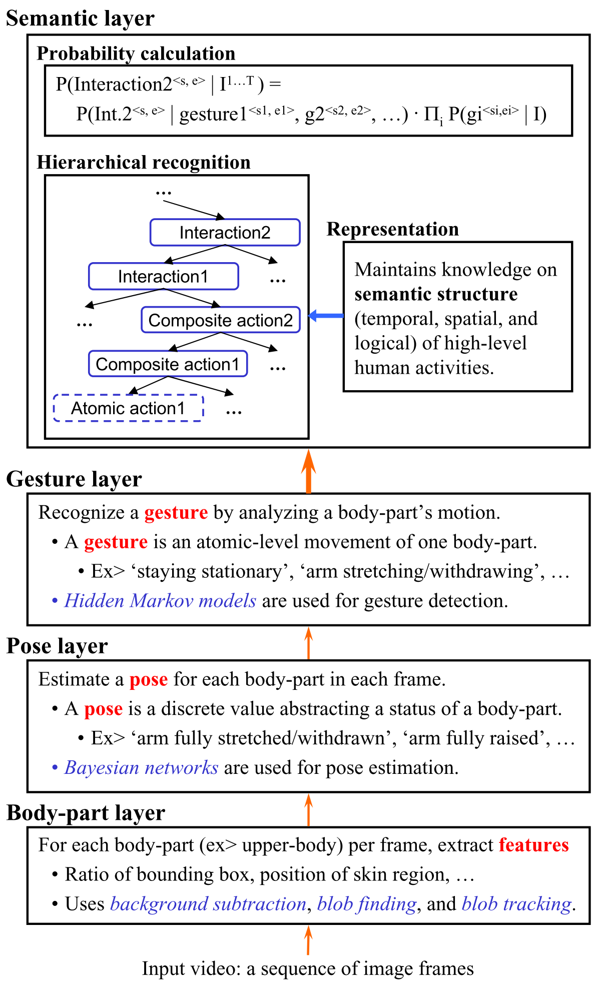

Semantic layer
Probability calculation
P(Interaction2<s, e> | I1…T) =
P(Int.2<s, e> | gesture1<s1, e1>, g2<s2, e2>, …) · Πi P(gi<si,ei> | I)
Hierarchical recognition
Interaction2
Interaction1
Composite action2
Composite action1
Atomic action1
Representation
Maintains knowledge on semantic structure (temporal, spatial, and logical) of high-level human activities.
Gesture layer
Recognize a gesture by analyzing a body-part's motion.
• A gesture is an atomic-level movement of one body-part.
• Ex> 'staying stationary', 'arm stretching/withdrawing', …
• Hidden Markov models are used for gesture detection.
Pose layer
Estimate a pose for each body-part in each frame.
• A pose is a discrete value abstracting a status of a body-part.
• Ex> 'arm fully stretched/withdrawn', 'arm fully raised', …
• Bayesian networks are used for pose estimation.
Body-part layer
For each body-part (ex> upper-body) per frame, extract features
• Ratio of bounding box, position of skin region, …
• Uses background subtraction, blob finding, and blob tracking.
Input video: a sequence of image frames

**Fig. 1** Figure illustrating the overall framework of the system

recognize complex high-level activities, raw pixel-level image sequences must be processed up to the atomic components in the low-level. In this paper, we adopt the framework developed by Park and Aggarwal (2004a, 2006) for the low-level of the system. Their system is able to reliably recognize gestures such as 'a person raising his/her arm' and 'a person's head turning left' using various computer vision techniques. Their framework to recognize gestures is composed of several layers: the body-part layer, the pose layer, and the gesture layer. The body-part layer, the lowest layer, estimates the numerical status of each body part per image frame. Taking those numerical values as parameters, the pose layer extracts poses for each frame. The gesture layer then generates sequences of gestures from given sequences of poses. A pose is the abstraction of the state of one body part, and a gesture is the abstraction of meaningful subsequence of those poses. There is a one-to-one correspondence between an occurring gestures and an atomic action. Various pixel-level techniques are used for the body-part extraction layer. Bayesian networks are used to implement the pose layer, and dynamic Bayesian networks (DBNs) are implemented for the gesture layer.

The high-level of the system (i.e. the semantic layer) recognizes human activities, from atomic-level actions to high-level interactions, by comparing detection results from the low-level with representations of activities it wants to recognize. The semantic layer is designed to maintain its knowledge on the temporal and spatial structure of a human activity as a 'representation'. Using the representation of the activity, the system hierarchically matches the representation with detected gestures, recognizing time intervals of occurring activities. In addition, the probability associated with the detected time interval is computed, by measuring the confidence of the matching and that of low-level detections. As a result, the system probabilistically detects time intervals of human activities, which are classified into three types depending on the structure of the representation: atomic actions, composite actions, and interactions.

If a human activity has no sub-events and can be recognized directly from low-level of the system, it is classified as an atomic action. If the activity has multiple sub-events (atomics actions and/or composite actions), but all sub-events are of the same person, we call it a composite action. Otherwise, if the activity has sub-events of more than one person, the activity is classified as an interaction. Since an interaction can always be decomposed into actions of two persons, all interactions have composite characteristics inherently. As shown in Fig. 1, a composite action can have simpler composite actions as its sub-events, and an interaction can have other interactions as its sub-events. This suggests that the maximum number of levels of hierarchy that our system recognizes is not fixed, unlike previous approaches (Hongeng et al. 2004; Nguyen et al. 2005; Park and Aggarwal 2004a, 2004b; Shi et al. 2004).

The intuition behind dividing the recognition process into two levels is to enable the system to integrate advantages of both statistical computer vision techniques and concepts from the field of artificial intelligence. Statistical computer vision methods are able to cope with noise while they are difficult to train, in the case of complicated activities. Instead of applying statistical computer vision methods (e.g. Bayesian networks and hidden Markov models) for the recognition of composite activities, our system uses statistical methods only for the recognition of human gestures. Data on gestures, such as 'stretching an arm' or 'withdrawing an arm', are relatively common, enabling the system to reliably recognize them even with noise. In the high-level of the system, we adopt concepts and predicates from Allen's temporal logic (Allen and Ferguson 1994), and represent human activities in terms of simpler sub-events in the format close to the full first-order logic. A probabilistic recognition algorithm is designed to take advantage of both the low-level detection results and the high-level representations.

## 4 Low-Level Processing of the System

### 4.1 Body-Part Layer

The objective of the body-part layer is in the estimation of features in each frame so that the pose layer can correctly estimate the pose of each body part. The body-part layer contains pixel-level, blob-level, and object-level processing to extract meaningful information from a sequence of raw images. We use a hierarchical methodology developed by Park and Aggarwal (2004a, 2006), in order to construct quantitative image features from one input frame. Their system parameterized the state of three body parts (head, upper-body, and lower-body) in terms of ellipses and convex hulls. Maintaining the overall structure of the system, we have reimplemented the framework to extract additional features for more reliable analysis of the status of body parts. Our new system explicitly tracks the hand position, which provides additional important features. The new system also considers the fact that skin blobs can merge during the interactions. As a result, the body-part layer of our system is more reliable compared to the previous system in tracking body-parts, extracting three features: ellipses of each body part, convex hulls of each body part, and the explicit hand position.

### 4.2 Pose Layer

In the pose layer, a pose for each body part is estimated based on features extracted by the system's body-part layer. A pose is the abstraction of the body part's static state in one image frame. For each image frame, the pose that

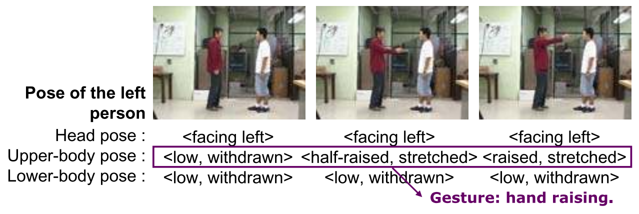

Pose of the left person
Head pose :
Upper-body pose :
Lower-body pose :
<facing left> <facing left> <facing left>
<low, withdrawn> <half-raised, stretched> <raised, stretched>
<low, withdrawn> <low, withdrawn> <low, withdrawn>
Gesture: hand raising.

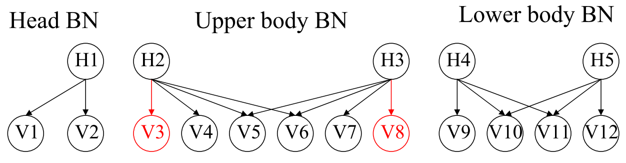
**Fig. 2** An example figure showing poses of each frame and the overall gesture. The left person is constantly raising his hand to point the other person

best describes the instantaneous configuration of the body part is selected based on parameters from the body-part layer. We constructed one-dimensional states for a head pose, describing the torso direction. Upper-body and lower-body poses have a two dimensional structure, each corresponding to vertical and horizontal positions of a hand and a leg. For example, assume that a person is standing still, facing left with his/her arm fully raised and stretched. Then, his/her head pose will be left, the upper-body pose will be $\langle \text{high}, \text{stretched} \rangle$, and the lower-body pose will be $\langle \text{low}, \text{withdrawn} \rangle$. Figure 2 shows example results of the pose estimation.

Extending Park and Aggarwal's (2004a) work, Bayesian networks are implemented to estimate a pose of each body part. The body-part parameters, estimated from the lower-layer, are converted into discrete values and are treated as observations produced by a specific pose. Bayesian networks estimate the pose for each frame, from the given observations and probabilities in the network. Once these Bayesian networks are pre-trained with appropriate training images with correct poses labeled by a human (i.e. supervised learning), they probabilistically estimates a pose corresponding to each body part per frame. Figure 3 illustrates the structure of the Bayesian networks and possible states for the pose of each body part, which is a final output of the pose layer. Note that new features, hand positions, are added: $V3$ and $V8$. As a result of the pose layer, an input image sequence is converted into a sequence of poses.

## 4.3 Gesture Layer

A gesture is an elementary movement of a body part. Taking the sequence of poses for each body part as input, the gesture layer detects possible gestures occurring along the sequence. Essentially, gestures are sub-sequences of a sequence of poses. The objective of the gesture layer is to recognize a set of all occurring gestures. Each occurring gesture has its starting time and ending time, which might overlap with other gestures. Example results of the gesture detection based on the pose estimation results are shown in Fig. 2.



H1: Head = {0:front, 1:left, 2:right, 3:rear}
H2: ArmV={0:high, 1:mid-high, 2:mid-low, 3:low}
H3: ArmH={0:withdrawn, 1:intermediate, 2:strecthed}
H4: LegV={0:high, 1:middle, 2:low}
H5: LegH={0:withdrawn, 1:intermediate, 2:strecthed}

V1: angle of vector from head to face
V2: ratio of face to head
V3: y-position of hand blob
V4: y-pos of far most point of upper body
V5: upper body ellipse ratio
V6: upper body ellipse rotation
V7: x-pos of far most point of upper body
V8: x-position of hand blob
V9: y-pos of far most point of lower body
V10: lower body ellipse ratio
V11: lower body ellipse rotation
V12: x-pos of far most point of lower body

**Fig. 3** Figure of the Bayesian network and explanation of meaning of nodes. The Bayesian network estimates the state of hidden nodes (i.e. poses), based on observations. The Bayesian network reduces dimensions from twelve into five

We construct hidden Markov models (HMMs) to detect the gestures occurring inside the sequence of frames. In order to recognize a sequence of gestures for each body part, we constructed one HMM per each gesture type. Types of gestures which our system is recognizing in this paper are similar to those in the paper presented by Park and Aggarwal (2004a). In addition to the gestures they recognized, we recognize 'start stretched', 'stay withdrawn', 'stay raised', and 'stay lowered' gestures for the 'upper-body' and 'lower-body'. As a result, the body part head has six possible gestures: 'keep facing front', 'keep facing back', 'keep facing left', 'keep facing right', 'turn left', and 'turn right'. The body part 'upper-body' has two separate dimensions of gestures, each corresponding to the horizontal movement and the vertical movement of the arm. Gestures 'stretch', 'withdraw', 'stay withdrawn', and 'stay stretched' correspond to the former case, and gestures 'raise', 'lower', 'stay raised', and 'stay lowered' correspond to the latter case. The gestures possible for the lower-body are exactly the same as that of the upper-body, except that now the subject of the gesture is the leg, not the arm. The structure of the HMMs is shown in Fig. 4.

The objective of the gesture layer is to detect which HMM created the observed sequence of poses and at what point. This is the traditional evaluation problem of the HMM. More specifically, the evaluation problem of the HMM is to determine the probability that a particular sequence of visible states, i.e. poses in our case, was generated by a corresponding model. In our gesture layer, each of the HMMs runs in parallel, measuring the likelihood of the corresponding gesture based on input. Additionally, for each body part, the 'noise HMM' is constructed to cover input sequences that are not related to any gesture we defined. The

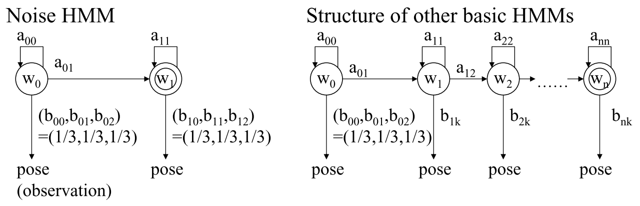


**Fig. 4** Structure of ‘noise HMM’ and other HMMs. For each gesture to be recognized, one HMM will be constructed in order to recognize corresponding gestures. Additionally, for each body part, one noise HMM will be created. Probability $a_{ij}$ corresponds to the transition probability from state $w_i$ to $w_j$. Probability $b_{jk}$ corresponds to the probability of observing $k$, when the real state of model is $w_j$

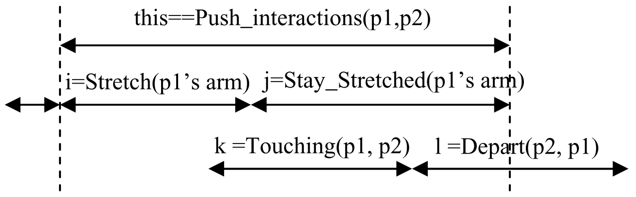


**Fig. 5** Example of necessary relationship among time intervals for interaction ‘push’

‘noise HMM’ tends to have the highest likelihood for meaningless sequences, making all gestures not to be detected.

These HMMs are pre-trained to generate observations corresponding to each gesture. The training videos with correct gesture labels attached to time intervals (i.e. a pair of a starting time and an ending time) are given to the HMMs, in order to estimate the transition and observation probabilities of the HMMs. Once the HMMs are trained, they are used by the system for the automated recognition of gestures.

We use the backward-looking forward algorithm to calculate the likelihood for each HMM. This works in the same way as a forward algorithm until detecting the ending point of the gesture. If the likelihood of a HMM exceeds the probability threshold at frame $t$, we assume that the gesture corresponding to the HMM occurred, and the ending time of that gesture is $t$. Once the ending time of the gesture is detected, then the algorithm runs a backward algorithm to find the starting point of the gesture. After detecting the starting time and ending time of the gesture correctly, the algorithm proceeds to frame $t + 1$. As a result of the gesture layer, a set of gestures labeled with starting and ending times is created for each body part. Input noises and miscalculation from lower layers are handled in this layer through HMM.

## 5 Semantic Layer: Representation

In the semantic layer, the system maintains representations of human activities which are encoded based on their semantic structure. The hidden Markov models presented in the previous section is able to recognize gestures such as ‘arm stretching’ and ‘arm withdrawing’, but HMMs themselves are not sufficient to recognize high-level interactions like ‘pushing’ and ‘hand shaking’. In order to recognize high-level activities based on gesture detections, we take a description-based approach in our semantic layer. That is, our system maintains the representation of an activity describing how gestures must be concatenated temporally, spatially, and logically in order to form the activity, and takes advantage of them for the recognition. In this section, we introduce the key concepts essential to represent activity structures, and present a formal representation syntax to describe activities using them. The recognition algorithm using the constructed activity representations will be discussed in the next section.

Our representation approach is a hierarchical approach, which decomposes an activity into several simpler activities called ‘sub-events’ and describes necessary relationships among the sub-events. We first present the concept of ‘time intervals’, which are associated with activities and their sub-events to represent their starting and ending times. Next, we introduce the ‘predicates’ used to formally describe temporal relations between time intervals, spatial relations between persons, and logical relations concatenating other relations. Finally, a programming language-like representation syntax is provided, which allows the description of structures of activities using time intervals and predicates.

### 5.1 Time Intervals

A time interval intuitively is the time associated with an occurring activity. The time intervals we discuss throughout this paper are always associated with designated actions or interactions that we are interested in. In Allen and Ferguson’s (1994) interval temporal logic, a time interval is defined in the linear time line, with a fixed starting point and ending point. Since activities are rarely instantaneous but take certain periods of time when occurring, the interval representation of time is appropriate for describing activities. Allen and Ferguson attempted to represent an event by presenting necessary conditions for the event’s time interval. Our system follows their approach, while representing hierarchical activities explicitly. We associated time intervals to the activity being represented and its sub-events, and represent the activities by presenting their temporal structure.

The intuition behind time interval associations is to describe necessary temporal structure of an activity in terms of the relations among intervals. Figure 5 shows the example time intervals of the human-human interaction ‘push’ and its sub-events. As shown in the figure, time intervals ‘$i$’, ‘$j$’, ‘$k$’, ‘$l$’, and ‘this’ are associated with activities, and the intervals are organized sequentially and concurrently. Each time intervals ‘$i$’, ‘$j$’, ‘$k$’, and ‘$l$’ corresponds to a simpler activity. The variable ‘this’ is a special variable, always indicating the time interval of the defining activity itself.

Figure 5 explicitly illustrates the temporal structure of the interaction ‘push’. In order for the interaction ‘push’ to occur, all of time intervals of its sub-events ‘$i$’, ‘$j$’, ‘$k$’, and ‘$l$’ has to be detected and they have to satisfy following temporal orders: (i) The time interval ‘this’, assigned for the pointing interaction itself, must start with time interval ‘$i$’ and finish with time interval ‘$j$’, each assigned for an arm stretching activity and a staying arm stretched activity of person1. (ii) Time intervals ‘$i$’ and ‘$j$’ must happen consecutively, and ‘$i$’ must be before ‘$j$’. In addition, (iii) time interval ‘$k$’ must occur during ‘this’, where ‘$k$’ indicates touching interaction of two persons. Finally, (iv) ‘$k$’ and ‘$l$’ must occur consecutively, suggesting that one person must be forced out as a consequence of the push.

We should note that time intervals of simpler activities are used when representing the relationships. This enables the system to use already defined activities to define new higher-level events, providing a concept of hierarchical activity representation. We denote all activities included in the relationships as ‘sub-events’ of defining activity. That is, activity ‘Stretch($p1$’s arm)’, ‘StayStretched($p1$’s arm)’, ‘Touching($p1$, $p2$)’, ‘Depart($p2$, $p1$)’ are sub-events of the activity ‘push interaction’ in this example.

### 5.1.1 *Special Time Interval ‘This’*

The concrete definition of the time interval ‘this’ is the key for hierarchical activity representation. Once the activity is correctly represented in terms of ‘this’ and other sub-events, the activity can be used as a sub-event of other higher-level activities. It is the ‘this’ which is providing robustness to our system compared to previous description-based recognition systems. In most of the previous description-based approaches, the time interval that includes time intervals of all of its sub-events is defined to be the resulting time interval of a represented activity (Hongeng et al. 2004; Siskind 2001; Vu et al. 2003). On the other hand, our system uses the special variable ‘this’ to define the resulting time interval. We already saw an example of the use of ‘this’ in Fig. 5.

There are several advantages to having a separate variable indicating the resulting time interval of a represented activity. First of all, as mentioned above, it is the special variable ‘this’ which is giving us the power of hierarchical representation. Secondly, our representation has freedom in setting the range of resulting time intervals. The ‘this’ can be defined within any range the user chooses. This generalization is useful when representing human activities with pre-conditions and/or effects. In addition, the variable ‘this’ enables us to represent an important class of high-level activities called ‘recursive activities’, which will be discussed in Sect. 7. Finally, our representation can be easily converted into the equivalent first-order logic, enabling the system to take advantage of research from the field of artificial intelligence.

## 5.2 Predicates

In the above example, all necessary conditions and relationships for the event are explained in English. Conceptually, the verbal description of time intervals and their relationships presented in the previous subsection is a valid form of the human activity representation. However, since it is not humans but the computer system that has to maintain the representation, we need a formal method of expressing the relationships. In order to provide a formal methodology to describe relationships, we present three categories of predicates in this subsection: temporal, spatial, and logical predicates. Temporal predicates express the relationship among time intervals of sub-events. Spatial predicates, on the other hand, describe the relationship between persons involved in the interactions. Logical predicates, ‘and’, ‘or’, and ‘not’, concatenate multiple temporal and spatial predicates to construct an overall representation for the event description.

### 5.2.1 *Temporal Predicates*

Temporal relationships are extremely important when describing human actions and interactions. Usually, actions and interactions of humans consist of sequences of sub-events. Temporal predicates not only provide us with a mechanism to define such sequential relations, but also help us to provide restricting conditions for the actions and interactions. We directly adopt the temporal relations among time intervals introduced in Allen’s interval temporal logic (Allen and Ferguson 1994). ‘before’, ‘meets’, ‘overlaps’, ‘starts’, ‘during’, and ‘finishes’ are the predicates defined in Allen’s interval temporal logic. Each predicate takes two time intervals as a parameter for the predicates, and decides whether they are true or false. Let $a$ and $b$ be the time intervals, $(a_{start}, a_{end})$ and $(b_{start}, b_{end})$.

$$\text{before}(a,b) \iff a_{end} < b_{start}$$
$$\text{meets}(a,b) \iff a_{end} = b_{start}$$
$$\text{overlaps}(a,b) \iff a_{start} < b_{start} < a_{end}$$
$$\text{starts}(a,b) \iff a_{start} = b_{start} \quad \text{and} \quad a_{end} < b_{end}$$
$$\text{during}(a,b) \iff a_{start} > b_{start} \quad \text{and} \quad a_{end} < b_{end}$$
$$\text{finishes}(a,b) \iff a_{end} = b_{end} \quad \text{and} \quad a_{start} > b_{start}$$

### 5.2.2 *Spatial Predicates*

Spatial predicates define the spatial relationship between two agents or objects. Thus, they can be defined only in terms of interactions. If any interaction contains some spatial predicates, and $t$ is the satisfying time interval of that event, those spatial predicates will always be true in the time interval $t$.

We designed two spatial predicates: 'near' and 'touch'. The 'near' predicate provides us with information on whether two persons are closer than a given relative distance value or not. The distance between two persons is divided by the mean of their heights, producing the relative distance. The 'touch' predicate is true if and only if the boundary ratio that two persons share is greater than the threshold parameter.

$$\text{near}(\text{person } i,\ \text{person } j,\ \text{threshold}) \iff$$
$$(\text{Relative distance between } i \text{ and } j) < \text{threshold}$$
$$\text{touch}(\text{person } i,\ \text{person } j,\ \text{threshold}) \iff$$
$$(\text{Overlapping boundary of } i \text{ and } j) > \text{threshold}$$

### *5.2.3 Logical Predicates*

Logical predicates include the 'and', 'or', and 'not' predicates. These are elementary logical predicates. All these predicates can take any relationship as a parameter. The 'and', 'or', and 'not' predicates are defined in an obvious manner. That is, logical predicates can concatenate temporal and spatial predicates to express relationships. The predicate 'and' holds if and only if relations described in all two parameters are satisfied. The predicate 'or' holds if more than one of two parameters is satisfied. We say that the 'not' of a relationship is satisfied if and only if the relationship parameter is false.

## 5.3 Representation

In this subsection, we present a methodology to construct a formal and machine-understandable representation of human activities. As we have discussed, the representation of an activity must consist of two essential components: the time intervals associated with the sub-events, and the necessary conditions needed among them. What we present in this subsection is a formal representation syntax, which enables the usage of the time intervals and the predicates to describe the semantics of the activities. We use context-free grammar (CFG) to describe the syntax of the representation, similar to that of programming languages. Following the syntax, we are able to construct machine-understandable descriptions of human activities. The constructed representations are maintained by the system for the recognition of the activities, which we will discuss in the next section.

We considered the hierarchical nature of human activities when constructing the representation. A human activity is described in terms of simpler activities, i.e. sub-events of the activity, which themselves might be composed of their own sub-events. We start constructing the representation by first defining the 'atomic actions', the elementary components of human actions not having any sub-events. Using atomic actions as basic building blocks, composite actions are represented next. Composite actions, once defined, can be treated as a sub-event of other composite actions, representing higher-level actions. Furthermore, actions of multiple persons can be concatenated to construct interactions, which themselves may be used as a sub-event of higher-level interactions. As a result, we are able to construct a human activity with any levels of hierarchy.

### *5.3.1 Atomic Action Representation*

Atomic actions are the most elementary component of human activities, which may not be divided into simpler meaningful movements. Atomic components of human actions and interactions are the gestures, recognized through system's lower-level. Therefore, we can construct one atomic action from one gesture. However, gestures solely are insufficient to represent the actions. In order to represent actions, the system needs to explicitly specify the subject and object of the actions. Following the linguistic theory of 'verb argument structure', we represent atomic actions as ⟨agent–motion–target⟩, adopting Park's operation triplet (Park and Aggarwal 2004a). Putting subject and object information together with the gesture, we construct the operation triplet. For example, 'person 1 stretched his hand to the person 2's head' is an atomic action, because only one gesture is involved in the action. Gesture 'stretch' is the motion of this atomic action. In the operation triplet, 'person 1's hand' is the agent and 'person 2's head' is the target.

### *5.3.2 Composite Action Representation*

If an action contains two or more atomic actions, it is classified as a composite action. Sub-events of composite actions can be atomic actions, or even other composite actions. The only constraint when constructing composite actions is that only the actions of the same person can become the sub-events. Otherwise, it becomes an interaction, rather than a composite action.

Figure 6 illustrates the example time intervals and their relationships of the composite action, 'shake-hands action'. The 'shake-hands action' represents an action that a person is performing in the hand shake interaction. That is, the person stretches his/her arm, stays stretched for some period, and then withdraws it. There are three sub-events participating in the 'shake-hands action': 'stretch', 'stay stretched', and 'withdraw'. Each sub-event has an associated time interval variable: '$x$', '$y$', and '$z$'. In addition, the figure describes that '$x$', '$y$', and '$z$' have to be done in a sequential order.

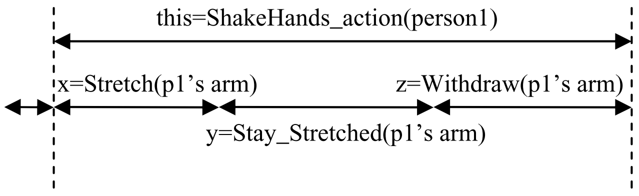


**Fig. 6** Example illustrating the atomic actions' time intervals and their relationships needed for the composite action, 'shake-hands action'

As we have discussed in Sect. 5.1, the figure description of temporal structure of the activity using time intervals is a sufficient form of representation since it is capturing necessary temporal relations among them. The problem is to convert this human-understandable conceptual representation into a machine-understandable programming language-like representation of the activity. Therefore, we construct the representation syntax that the user should follow when encoding the representation.

The representation for composite actions consists of two parts: a list of variables corresponding to time intervals associated with designated sub-events, and the relationships among those variables. The first component can be represented as a list of (time interval, sub-event) pairs. The second component, which represents necessary conditions for composite actions, is defined through predicates mentioned in Sect. 5.2. Variables that are defined and the special variable 'this', representing the defining action itself, are used in order to specify the relationships. Therefore, we are able to represent a composite action in terms of the relationship between 'this' and other time interval variables '$t1$', '$t2$', ..., which are time intervals of sub-events.

As a format of the representation syntax, we use a context-free grammar (CFG). CFG naturally leads the representation to use concepts recursively, enabling the action to be defined based on sub-events. We emphasize once more that our CFG-based representation scheme is a formal syntax to generate the representation, not a production rule to analyze the input images. This differentiates our work from the work done by Ivanov and Bobick (2000) and other similar conventional uses (i.e. syntactic approached presented in Sect. 2.2) of CFG for syntactic recognition processes.

Our CFG does not generate sequences of poses or gestures directly. Rather, we construct a representation of composite actions using the CFG. A representation built through the CFG describes all participating sub-events, and their relationships. Sub-events can either be atomic actions or other already represented composite actions. Even though the CFG does not create the sequences of poses or gestures directly, we will be able to recognize composite actions through detecting sequences that satisfy the representation constructed with our CFG. With our CFG, we are able to represent any action if its temporal, spatial, and logical structure can be described in terms of the predicates we have defined.

The full representation syntax is presented in Appendix A.2. This is similar to the syntax of programming languages (note that syntax for *C* and *Java* is in CFG). Following the syntax, we are able to represent semantics of an activity by specifying sub-events composing the activity, associating time interval variables with the sub-events, and describing necessary relationships using the predicates.

For example, let's look into the composite action 'shake-hands action' again. As we informally defined previously in Fig. 6, we associate variables '$x$', '$y$', and '$z$' with sub-events 'stretch', 'stay stretched', and 'withdraw'. Then, relationships are represented in terms of predicates: meets($x$, $y$), meets($y$, $z$), starts($x$, this), and finishes($z$, this). Therefore, the formal representation of 'shake-hands action' is defined through our CFG scheme as follows.

```
StretchHand(i) = atomic_action
    (⟨person i's hand, stretch, other person⟩);
StayStretchedHand(i) = atomic_action
    (⟨person i's hand, stay stretched, other person's hand⟩);
WithdrawHand(i) = atomic_action
    (⟨person i's hand, withdraw, null⟩);

ShakeHandsAction(i) = (
    list( def(x, StretchHand(i)),
          list( def(y, StayStretchedHand(i)),
                def(z, WithdrawHand(i) ) )
    ),
    and( meets(x, y),
         and( meets(y, z),
              and(starts(x, this), finishes(z, this) ) )
    )
);
```

### 5.3.3 *Interaction Representation*

Interactions are composed of the actions and/or interactions of two persons. In the case of actions, actions are classified into atomic actions and composite actions. On the other hand, all interactions have composite characteristics. Therefore, except for the fact that sub-events of interactions can be actions of both persons, the CFG production rule, i.e. representation scheme, of interactions is almost identical to that of composite actions. Further, spatial predicates also can be used to describe relationships for interactions. The full CFG syntax is presented in Appendix A.2.

The following example shows how a 'hand-shake' interaction can be represented by following our CFG syntax. The already defined composite action, 'shake-hands action' of two persons, is used as a sub-event of the interaction 'shake-hands interaction'. If person $i$ and $j$ do the action 'shake-hands action' concurrently, and their hands touch, we regard it as a hand shake interaction.

```
TouchingInteraction(i, j) = (null, touch(i, j, 0));
ShakeHandsInteractions(i, j) = (
    list( def(x, ShakeHandsAction(i)),
          list( def(y, ShakeHandsAction(j)),
                def(z, TouchingInteraction(i, j)) )
    ),
    and( and( during(z, x), during(z, y)),
         and( starts(z, this), finishes(z, this) )
    )
);
```

## 6 Semantic Layer: Recognition

The semantic layer recognizes human activities by analyzing whether its sub-event detection results satisfy the representation of the activity it wants to recognize. Atomic actions are directly recognized using the gesture recognition results from the low-level of the system. Composite actions are recognized based on the recognition result of atomic actions and other composite actions. In the recognition of interactions, recognition results of any simpler activities, including other interactions, can be used. In the case of composite actions and interactions, recognition of activities is done by finding time intervals that satisfy all temporal and spatial relationships needed for the special variable 'this'.

### 6.1 Atomic Action Recognition

An atomic action is represented in terms of an operation triplet. By definition, an occurring time interval of an atomic action is that of a gesture specified through the *motion* term in the operation triplet ⟨agent–motion–target⟩. If the gesture layer recognized a gesture specified in motion term of the triplet, and its subject and object corresponds to the agent and target term of operation triplet, the system concludes that the atomic action is recognized in that time interval.

### 6.2 Composite Activity Recognition

As we have presented in the previous sections, a representation of a composite action or an interaction has two components: time interval variable definitions and their relationship descriptions. Therefore, the recognition of composite actions and interactions requires two steps. In the first step, the system must recognize all of its sub-events. In the second step, the system needs to check whether the time intervals detected for the sub-events satisfy the relationships described in the activity representation. If they do, the system can safely deduce that the activity occurred with the sub-events detected. However, this is not a trivial process since an activity described to be a sub-event of another activity may occur multiple times. Among multiple occurrences of the sub-event, the system must decide which among them (if there is any) contributed to the occurrence of the activity. That is, each variable has multiple possible time interval assignments, and the system must choose the correct one to be assigned and check its relationships in order to recognize the activity.

Our system searches for a combination of time interval assignments that satisfies relationships needed for the activity. If there are $n$ variables and $m_1, m_2, \dots, m_n$ number of time intervals for each variable, then there exist $\prod_{i=0\ to\ n} m_i$ possible combinations of (variable, time interval) pairs. In addition, the special variable 'this' can be associated with $T^2$ number of time intervals where $T$ is the total number of frames. The goal is to find the combination among $T^2 \cdot \prod_{i=0\ to\ n} m_i$ of them, which satisfies the relationships specified in the representation. This is a traditional constraint satisfaction problem. The system must find a specific combination of (variable, time interval) pairs that satisfies relationships, among all possible combinations.

We have developed a heuristic methodology to approximate the correct solution in a polynomial amount of computation. Originally, the constraint satisfaction problem is a NP-hard problem, which requires $O(T^2 \cdot \prod_{i=0\ to\ n} m_i)$ time complexity in our case. One of commonly used approximation techniques is to reduce the number of candidate assignments, $m_i$. That is, instead of searching for all possible assignments, we may search for time intervals only within a certain window at each time point. The system requires to repeatedly search for the correct combination after each time frame, but this reduces the number of candidate time intervals per variable dramatically.

The extreme case of this is to make the system only use the most recent time interval per variable. This simplification always gives the correct result with the assumption that 'one sub-event occurs only once during an execution of a human activity'. In practice, this is a good assumption which is true in most cases. Therefore, at each time point, there exists only one combination to check its relationships. In addition, once a combination has been found, the time interval for 'this' can be calculated by narrowing its range based on other detected time intervals. For example, assume that the time interval for '$x$' is (1, 4) and that for '$z$' is (10, 15) in case of 'shake-hands' (Fig. 6). The time interval for 'this' has to be (1, 15), since it has to start with '$x$' and has to end with '$z$'. As a result, we have developed a recognition algorithm which only requires the time complexity of $O(rT)$, where $r$ is the number of relationships per action.

Since our representation for actions and interactions has a hierarchical structure, i.e. one action or interaction has multiple sub-events which may be decomposable as well, our action and interaction recognition is done in a hierarchical way. If a composite action or an interaction A has actions B and C in its variable list, i.e. B and C are sub-events

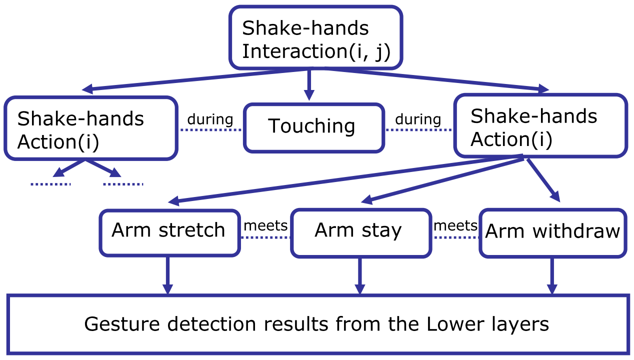


**Fig. 7** Example recognition process tree of the interaction 'shake-hands interaction'

of A, then the recognition system first recognizes action B and C. If B and C are composite themselves, they again trigger recognition of their sub-events in the variable list. At some point, all the sub-events will be atomic actions, which the system recognizes using the algorithm described in the previous subsection. This is similar to tree traversal where activities are nodes, variable lists specify edges, and atomic actions are leaves. In order to recognize the root action or interaction, the system must recognize its child. This process continues until the system reaches the leaves. Once the system reaches leaves, the system is able to compute time intervals of composite actions or interactions that have atomic actions as sub-events. The system traverses back to the root, recognizing all internal nodes from leaves to the root. At each internal node, the system has to solve the constraint satisfaction problem that we have discussed above. Figure 7 illustrates the recognition process of the 'shake-hands interaction'.

## 7 Recursive Activities

In previous sections, we represented human activities in terms of a strictly fixed number of sub-events. However, for abstract high-level human activities such as 'fighting' or 'greeting', the number of sub-events is not clear. We cannot say that the person punching the other three times is a fighting activity while the person punching four times is not. Rather, those high-level human activities tend to have recursive characteristics. Assume that the system detected a fighting interaction in some time interval. If another punching interaction is directly followed by a detected fighting interaction, the system must detect a longer fighting interaction covering the latest punching, based on the detection of the shorter fighting interaction.

### 7.1 Recursive Activity Representation

Here we provide the concept of recursive representation for human activities. In the case of recursive actions and interactions, a defining human activity can become sub-events of itself. We do not describe the actual overall length of the defining human activity, but instead use a simpler (or shorter) identical activity as a sub-event. This recursive representation is able to catch the varying length of the defining activity, since the level of hierarchy can grow indefinitely.

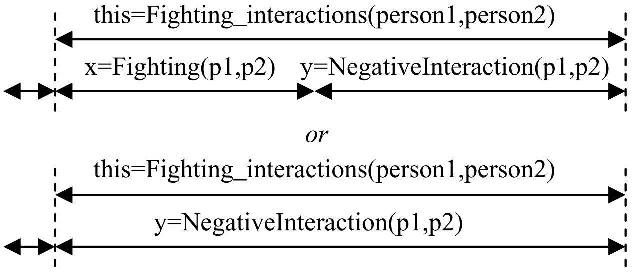

this=Fighting_interactions(person1,person2)
x=Fighting(p1,p2)
y=NegativeInteraction(p1,p2)
or
this=Fighting_interactions(person1,person2)
y=NegativeInteraction(p1,p2)

**Fig. 8** The necessary temporal relationship among time intervals for recursive interaction 'fighting'. The base case is shown in *bottom*

Essentially, the syntax provided in the CFG-based representation scheme is able to describe recursive activities without any modification. Modification is not needed for the syntax, but for the interpretation in the representation scheme. Previously, the InteractionName$(i, j)$ in the CFG syntax (Appendix A.2) can only denote activity names (strings of characters) that had been defined already. That is, only an activity that has been defined may be used as a sub-event. If we modify the interpretation of InteractionName$(i, j)$ a little bit, enabling the user to use the name of the defining activity also, the recursive activities can be represented easily.

Another important characteristic of recursive activities is the existence of the base case. Recursive activities are recognized by detecting the simpler identical activity. Therefore, at some point, the system needs the seed (or core) activity of the detection, which does not rely on the detection of a simpler identical activity. This is called a base case of the recursive activity. When representing the recursive activity, the user must always construct the base case of the activity. Otherwise the system will fail to understand the activity.

Let's look at the actual 'fighting' interaction for example. In principle, the interaction 'fighting' is defined as a concatenation of a simpler 'fighting' and one 'negative interaction' as illustrated in Fig. 8. The 'negative interaction' is disjunctive concatenation (i.e. 'or' product) of 'punch', 'kick', and 'push', which are composing activities of the fighting. The base case of the 'fighting' is one 'negative interaction'. The 'greeting' interaction can be represented in a similar manner. In the case of the 'greeting' interaction, the interaction must contain at least one 'shake hands' interaction which is a core of 'greeting'. Thus, the base case of the greeting is the single hand shaking interaction. The following shows the formal representation for 'negative interaction', 'fighting', and 'greeting'.

```
NegativeInteraction(i, j) = (
    list(  def(x, PunchInteraction(i, j)),
           list(  def(y, KickInteraction(i, j),
                  def(z, PushInteraction(i, j))))
    ),
    or(    equals(this, x),
           or(equals(this, y), equals(this, z))
    )
);
FightingInteraction(i, j) = (
    list(  def(x, FightingInteraction(i, j)),
           def(y, NegativeInteraction(i, j))
    ),
    or(    equals(y, this),
           and(  meets(x, y),
                 and(  starts(x, this), finishes(y, this)))
    )
);
GreetingInteraction(i, j) = (
    list(  def(x, ShakeHandsInteraction(i, j)),
           list(  def(y, PositiveInteraction(i, j)),
                  def(z, GreetingInteraction(i, j)))
    ),
    or(    equals(x, this),
           or(    and(  meets(y, z),
                        and(  starts(y, this),
                              finishes(z, this))),
                  and(  meets(z, y),
                        and(  starts(z, this),
                              finishes(y, this))))
    )
);
```

### 7.2 Recursive activity recognition

For the recognition of recursively described actions and interactions, an iterative approach is used. We explain this iterative algorithm with the example, the ‘fighting’ interaction. The system starts with setting time interval ‘$x$’, corresponding to sub-event ‘fighting’, to null. Then, the system is only able to find base cases. In the case of ‘fighting’, a single ‘negative interaction’ corresponds to a base case. Once we found some initial ‘fighting’ interactions, we now treat those detected ‘fighting’ as sub-events of the 2nd iteration. The ‘fighting’ interactions found through the 2nd iteration serve as a sub-event of the 3rd iteration. Iteration continues until no ‘fighting’ is detected additionally. The detection result of nth iteration serves as a sub-event of $(n+1)$th iteration. With this iterative algorithm, given the detection result of the 1st iteration, i.e. the base case detection, we are able to detect recursive activities. The pseudo-code of the complete algorithm is provided in Fig. 9. The function CSP implies that we are solving the constraint satisfaction problem mentioned in Sect. 6.2. The system finds the occurring activity by checking all possible combinations of (variable, time interval) pairs, whether they satisfy the activity’s representation or not.

```
Detect(interaction i) {
  if (i is non-recursive) {
    for i's (v=variable, j=sub-event){
      add (v, Detect(j)) to the list;
      result = CSP(list, i);
    }
  }
  else result = RecursiveDetect(i);
  return result;
}

RecursiveDetect(interaction i) {
  let x denote the recursive sub-event;
  for i's (v=variable-x, j=sub-event){
    add (v, Detect(j)) to the list;
    result = CSP(list, i);
  }
  list.x = null;
  do {
    result = CSP(list, i);
    list.x = Union(current, result);
  } while (result!=current);
}
```

**Fig. 9** The algorithm for the recognition of recursive activities

## 8 Probabilistic Recognition

In this section, we introduce the methodology for the probabilistic recognition of human activities. First, in Sect. 8.1, we present a method to probabilistically recognize an activity when time intervals of all gestures composing the activity are provided together with their confidence probabilities from the gesture layer. Even when few gestures composing a high-level activity are detected with low confidence, this method gives the system an ability to recognize the activity if the confidence of the detected gestures are mostly high.

Next, in Sect. 8.2, the concept of ‘hallucination’ time intervals is introduced, which enables the system to recognize activities even with the complete failures of gesture detections (i.e. the situation where no correct time interval is detected by the gesture layer at all) by hallucinating the missing time intervals. Since our system models hallucinated gestures as ‘liquid’ activities in order to handle them efficiently, we present a fast algorithm to recognize high-level activities from liquid sub-events in Sect. 8.3. Liquid activities are activities that have a continuous nature, ‘touching’ for example. Hallucinations and liquid activities will be defined more formally in Sect. 8.2, and the detailed algorithm to recognize activities with them will be presented in Sect. 8.3.

## 8.1 Probability Calculation

Our probabilistic recognition algorithm consists of two parts: the time interval detection procedure and the probability calculation procedure for each time interval detected.

First, using the algorithm presented in Sect. 6.2, time intervals associated with a high-level activity can be detected hierarchically based on time intervals of detected gestures. However, unlike in the case of the deterministic recognition, even a time interval of a gesture detected with a small probability is now considered a valid candidate time interval in the case of probabilistic recognition. The gesture layer is asked to generate time intervals with the confidence probabilities attached to them, whenever the probability of a gesture occurring reaches any local maximum (i.e. the likelihood of the HMM reaches a local maximum). Except for the fact that more time interval candidates are associated with each gesture, generating more time intervals for atomic actions, the overall algorithm is equivalent. As a result, a time interval associated with a high-level activity is detected, and time intervals of gestures composing the activity is identified.

Once time intervals of an activity being recognized are detected, the system then computes the probability (or confidence) of the activity by considering probabilities associated with the detected gestures composing the activity. By avoiding deterministic decisions to be made by the gesture layer, our probabilistic algorithm is able to recognize human activities more reliably and accurately. The objective is to calculate the probability of an activity occurring in one time interval, given the sequence of images. If we denote images from frame 1 to $T$ as $I^T$, then the conditional probability of the activity $R$ occurred in the time interval $\langle s, e\rangle$ can be expressed as $P(R^{\langle s,e\rangle} \mid I^T)$. The goal of our algorithm is to calculate $P(R^{\langle s,e\rangle} \mid I^T)$ based on the recognition results of gestures, $P(G_i^{\langle s_i,e_i\rangle} \mid I^T)$ where $G_i$ is the $i$th gesture composing the activity.

In order to calculate probability of a high-level activity, we use the dependency information between the activity and its sub-events. The hierarchy tree (e.g. Fig. 7) illustrates dependencies among the activities similar to the Bayesian network. Activities associated with child nodes depend on the activity associated with a parent node. By the definition of the operation triplets, the gesture specified in an operation triplet depends on that atomic action, i.e. the leaf node. The main difference between the dependency among nodes in the hierarchy tree and those in the Bayesian network is that siblings of the hierarchy tree are not conditionally independent given the parent node; sub-events tend to occur together, implying that they are highly correlated.

We denote a union of sub-events of each element in set $S$ as $\mathrm{sub}(S)$. When an element $a$ of the set $S$ does not have any sub-events, the $\mathrm{sub}(S)$ is defined to be $\mathrm{sub}(S-a) \cup a$. Then, the probability $P(R^{\langle s,e\rangle} \mid I^T)$ can be enumerated using the dependency among nodes, as follows:

$$\begin{aligned}
&P(R^{\langle s,e\rangle}|I^T)\\
&\quad= P\big(\{R\}|\mathrm{sub}(\{R\})\big) \times P\big(\mathrm{sub}(\{R\})|\mathrm{sub}\big(\mathrm{sub}(\{R\})\big)\big)\\
&\qquad \times \cdots \times P\big(\mathrm{sub}^d(\{R\})|I^T\big)\\
&\quad= \prod_{i=0}^{d-1} P\big(\mathrm{sub}^i(\{R\})|\mathrm{sub}^{i+1}(\{R\})\big) \times P\big(\mathrm{sub}^d(\{R\})|I^T\big)\\
&\quad= \prod_{i=0}^{d-1} P\big(\mathrm{sub}^i(\{R\})|\mathrm{sub}^{i+1}(\{R\})\big) \times P\big(a_1,\dots,a_n|I^T\big)
\end{aligned} \tag{1}$$

where $a_1, a_2, \dots, a_n$ are leaf nodes (i.e. atomic actions) of the tree and $d$ is the depth of the tree.

Because of the characteristics of our representation, we can safely assume that an activity occurs if and only if all of its sub-events occur. That is, for all set of siblings $S$ in the tree, $P(S|\mathrm{sub}(S)) = 1$.

Therefore, $P(R^{\langle s,e\rangle} \mid I^T)$ can be simplified into the product of conditional probabilities among atomic actions and gestures. If we assume conditional independence among recognitions made by the gesture layer, the probability can be enumerated as follows:

$$\begin{aligned}
&P(R^{\langle s,e\rangle}|I^T)\\
&\quad= \prod_{i=0}^{d-1} P\big(\mathrm{sub}^i(\{R\})|\mathrm{sub}^{i+1}(\{R\})\big) \times P\big(a_1,\dots,a_n|I^T\big)\\
&\quad= 1 \times P\big(a_1,\dots,a_n|I^T\big)\\
&\quad= \sum_{g_1}\cdots\sum_{g_n}\big[P\big(a_1,\dots,a_n|g_1,\dots,g_n\big)\\
&\qquad \times P\big(g_1,\dots,g_n|I^T\big)\big]
\end{aligned} \tag{2}$$

We estimate the probability $P(a_1,\dots,a_n|g_1,\dots,g_n)$ using the regression techniques with binary features $g_1$, $g_2,\dots,g_n$. Both the linear regression with a minimum and maximum value of 0 and 1 and the logistic regression have been applied to estimate the probability value $P(a_1,\dots,a_n|g_1,\dots,g_n)$, resulting a similar outputs. In the case of the logistic regression, we assume the $P(a_1,\dots,a_n|$ $g_1,\dots,g_n)$ to be a logistic function of $g_1,\dots,g_n$:

$$P\big(a_1,\dots,a_n|g_1,\dots,g_n\big) = 1/\big(1 + e^{-\beta_0-\beta_1 g_1-\cdots-\beta_n g_n}\big) \tag{3}$$

where we estimate parameters $\beta_0,\dots,\beta_n$ through training.

In contrast to the training full conditional probability table that has $O(2^n)$ cases even for binary features, only $O(n)$ parameters need to be trained for the linear and logistic regressions. As a result, the high-level of the system can be trained with less training data, while appropriately estimating the real probability distribution.

After calculating the probability associated with an activity's time interval, i.e. $P(R^{\langle s,e\rangle} \mid I^T)$, the system can make a recognition decision by accepting time intervals with the probability above a certain threshold probability value and rejecting others. That is, only the time intervals with high confidence probability are recognized. Because of the characteristics of the linear (or logistic) regression, the probability of an activity tends to be high when one or more gestures have low confidence probability but all the others have high enough probability.

Recursive activities can also be recognized probabilistically using the method presented in this sub-section. Gestures composing a recursive activity can be revealed using the algorithm presented in Sect. 7.2, and the regression technique can be applied in a similar way to estimate the probability, theoretically. One practical problem arises when calculating the probability of recursive activities is on the number of training examples needed. Because of the characteristics of the recursive activities, the number of gestures composing the activity is not limited. For example, the number of gestures is six if the fighting is composed of three punching actions, while it is ten if the fighting is composed of five punching actions. In order for the regression technique to estimate the probability correctly, it must be trained with sufficient examples for each case. Theoretically, they can simply be trained with large training data. In practice, we usually limit the size of the recursive activities, since the amount of data that can be obtained is limited.

## 8.2 Concepts of Hallucinations and Liquid Activities

In this sub-section and the following sub-section, we present a methodology for the semantic layer to overcome complete failures of the gesture layer. For reliable recognition of high-level activities, the system must be able to detect occurring activities even when time intervals associated with one or few gestures are not detected because of the failure. The probability calculation method presented in the previous sub-section requires all gestures composing an activity to have associated time intervals at least with a low probability in order for it to recognize the activity. Therefore, we introduce the concept of 'hallucination' time intervals, which is inserted in the place of missing gestures with an extremely low probability. With hallucinated time intervals of gestures, our approach is able to recognize high-level activities even with imperfect lower layers probabilistically.

'Hallucinations' are time intervals which are inserted regardless of the gesture recognition results, to compensate for the failures of the gesture layer. Our usage of hallucinations is similar to that in previous syntactic approaches (Minnen et al. 2003). The difference is in the locations where hallucinations must be inserted, and the algorithm to recognize activities with concurrent hallucinations. Since syntactic approaches focus on the recognition of sequential activities, inserting hallucinations between correctly detected gestures was sufficient for them. However, our approach is designed not only to recognize sequential activities but also to recognize activities composed of complex concurrent sub-events. That is, in our representation, gestures may occur in time intervals with any starting time and any ending time, implying that the ability to insert hallucinations in all possible intervals is required. In principle, the system may put hallucination time intervals with all possible starting time and ending time, generating $T^2$ number of intervals where $T$ is the number of observed frames. However, maintaining $T^2$ number of time intervals for each gesture results in an exponential amount of computation for recognitions (more specifically, $O(nT^{2n})$ where n is the number of gestures composing the activity).

In order to model hallucinations concisely, we introduce the concept of 'liquid' and 'semi-liquid' time intervals. Siskind (2001) introduced the concept of liquid and semi-liquid events. Occurring time intervals of those activities are modeled with a range of starting time and a range of ending time. A liquid (or semi-liquid) time interval represents all time intervals whose starting time is within the specified range of it and whose ending time is within its range. If the range of its starting time and ending time are identical, the time interval is called a 'liquid'. Otherwise, it is called a 'semi-liquid'. The human activity 'touching' is a good example of an activity which can efficiently be modeled using a liquid time interval. If two persons are touching from time frame 5 to time frame 15, two persons are essentially touching in any time interval whose starting and ending are within (Hongeng et al. 2004; Park and Aggarwal 2004b).

We model a hallucinated time interval of a gesture as a liquid interval whose range of starting time and ending time is from 0 to $T$. Therefore, only one liquid time interval needs to be maintained as a hallucination for each gesture (instead of maintaining $T^2$ intervals).

## 8.3 Recognition of Liquid Activities

We construct an efficient algorithm to recognize human activities when one or more sub-event composing them are liquid time intervals. More specifically, a linear time algorithm is developed to check whether given combinations of sub-events' time intervals (including liquid time intervals) satisfy the specified temporal relationship or not. The process of the algorithm is as follows. First, the algorithm converts the formal representation of the human activity into a directed graph form, similar to Ryoo and Aggarwal (2007). Next, the algorithm associates a range of time for each node in the graph, and checks whether assigning an integer value within the range to each node is possible or not. If possible, it suggests that there exists at least one valid time interval combination that satisfies the temporal constraints.

### *8.3.1 Conversion into a Directed Graph Representation*

In our directed graph representation, a vertex is a time point (either starting time or ending time of a sub-event), and an edge from vertex $t1$ to vertex $t2$ implies $t1 < t2$. The purpose of this conversion is to calculate the necessary temporal ordering between times associated with an activity's sub-events. The procedure to convert our programming language-like representation into a directed graph representation is presented below.

First, the system must convert Allen's temporal predicates for time intervals into equalities and inequalities among time points. In our programming language-like representation, temporal relationships are specified as a logical formula of Allen's temporal predicates. Following the definition of temporal predicates, the representation can be converted into equalities and inequalities among time points as follows:

Let $a$ and $b$ be the time intervals, $(a_{start}, a_{end})$ and $(b_{start}, b_{end})$.

$$\text{before}(a,b) \implies a_{end} < b_{start}$$

$$\text{meets}(a,b) \implies a_{end} = b_{start}$$

$$\text{overlaps}(a,b) \implies a_{start} < b_{start} < a_{end}$$

$$\text{starts}(a,b) \implies a_{start} = b_{start} \quad \text{and} \quad a_{end} < b_{end}$$

$$\text{during}(a,b) \implies a_{start} > b_{start} \quad \text{and} \quad a_{end} < b_{end}$$

$$\text{finishes}(a,b) \implies a_{end} = b_{end} \quad \text{and} \quad a_{start} > b_{start}$$

Also, we add one trivial inequality $a_{start} < a_{end}$ for all time intervals $a$. As a result, logical concatenations of Allen's temporal predicates are converted into logical concatenations of equalities and inequalities among time points. Next, the system removes the predicate 'not', as follows. First, the system applies De Morgan's laws to enumerate the inequalities to make 'not's are only attached to a single predicate. Then,

$$\text{not}(t1 < t2) \implies t2 < t1 \quad \text{or} \quad t1 = t2$$

$$\text{not}(t1 = t2) \implies t1 < t2 \quad \text{or} \quad t2 < t1$$

The system then converts a logical formula into a disjunctive normal form (DNF). The end product is the disjunction of conjunctive clauses of pure equalities and inequalities. This suggests that the activity representation can be divided into several conjunctive clauses, where each clause presents necessary temporal conditions for the activity. There is no semantic difference between the DNF representation and the directed graph representation we plan to construct. For each clause, we formulate one directed graph to help the user visualize the representation. The system first calculates time points that are equal, checking the equalities in a clause. The system assigns one vertex for a set of time points which are equal. For example, if $t1 = t2$ and $t2 = t3$, only one vertex is assigned for a set $\{t1, t2, t3\}$. Then, an edge is constructed from a vertex $v1$ to a vertex $v2$, if $\exists t1, t2$ such that $(t1 \in v1$ and $t2 \in v2$ and $t1 < t2)$. What this directed graph representation suggests is that the activity can be completed if and only if an integer value is correctly assigned for each vertex while satisfying the temporal order of the graph.

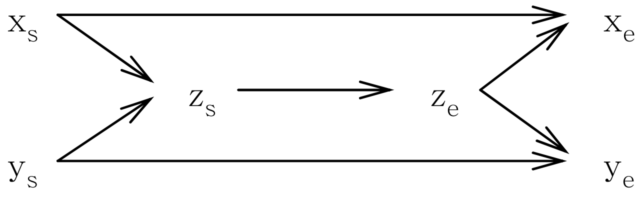


**Fig. 10** The directed graph representation of 'shake-hands'. $x$, $y$, and $z$ are time intervals of sub-events, each associated with ShakeHandAction$(i)$, ShakeHandAction$(j)$, and TouchingInteraction$(i, j)$ as illustrated in Sect. 5.3.3

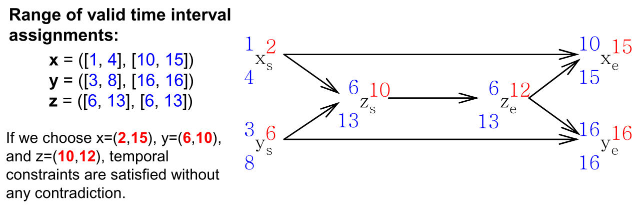
For example, the representation of the human-human interaction 'shake-hands' can be converted into a directed graph form as follows. The formal representation of it is presented in Sect. 5.3. First, the temporal predicates are converted into equalities and inequalities.

$$(x_{start} < z_{start} \text{ and } y_{start} < z_{start} \text{ and}$$
$$z_{end} < x_{end} \text{ and } z_{end} < y_{end} \text{ and}$$
$$x_{start} < x_{end} \text{ and } y_{start} < y_{end} \text{ and } z_{start} < z_{end}$$
$$\text{this}_{start} = z_{start} \text{ and } \text{this}_{end} = z_{end})$$

This already is a DNF composed of only one conjunctive clause. Figure 10 shows a final directed graph representation of the example.

### *8.3.2 Temporal Constraint Verification*

Once the direct graph is constructed, we associate the range of time to each node based on time intervals detected for sub-events. Next, we try to detect any contradiction caused by the association. Each node corresponds to either the starting time or ending time of a sub-event, whose range is specified based on liquid time intervals. For each given time range association, the system must verify that there exists at least one value in the assigned range for each node which does not result any temporal contradiction. In our temporal graph, the value of an ancestor node must be strictly less than its descendants. That is, since our system keeps track of time in a frame-based fashion (i.e. a time points is an integer value), the value of a child node must be greater than or equal to the (value of its parent) + 1. Figure 11 illustrates an example of valid integer value assignments that satisfy the temporal constraints.


**Range of valid time interval assignments:**

**x** = ([1, 4], [10, 15])
**y** = ([3, 8], [16, 16])
**z** = ([6, 13], [6, 13])

If we choose x=(**2**,**15**), y=(**6**,**10**), and z=(**10**,**12**), temporal constraints are satisfied without any contradiction.

**Fig. 11** (Color online) The directed graph representation of 'shake-hands' with ranges for time intervals associated. The two *blue* numbers on the *left* of each node show the minimum and maximum value of a valid time assignment. The *red* numbers on the *right* of the nodes are time values assigned within the range, which satisfy the temporal constraints of the graph without any contradiction. The goal of the algorithm is to check whether there exists such assignment when a range for each node is given

Therefore, the following constraint is posed for a time range association to verify that the association avoids the temporal contradiction of the activity.

$$\forall t1, t2 : (t2.\text{range}_{max} - t1.\text{range}_{min}) \geq \text{distance}(t1, t2)$$

where $t2$ is a descendant of $t1$.

This condition can be checked for each time range assignment in linear time using an algorithm similar to the tree traversal algorithm. Figure 12 shows the pseudo code of this linear time algorithm. In addition, after verifying that the condition is satisfied, the system can also compute a desired range for the time interval 'this' as follows:

$$\text{this}_{(start\ or\ end)}.\text{range}_{min}$$
$$= \max_{t1}(\text{distance}(t1, \text{this}) + t1.\text{range}_{min})$$
$$\text{this}_{(start\ or\ end)}.\text{range}_{max}$$
$$= \min_{t2}(t2.\text{range}_{max} - \text{distance}(\text{this}, t2))$$

where $t1$ is an ancestor of $\text{this}_{(start\ or\ end)}$ and $t2$ is a descent of $\text{this}_{(start\ or\ end)}$, and $\text{this}_{(start\ or\ end)}$ can either be a starting time or ending time of the time interval 'this'.

Overall, the algorithm to recognize human activities based on liquid time intervals is developed. As a result, high-level activities can be recognized hierarchically even with imperfect gesture recognition by modeling hallucinations as liquid time intervals.

## 9 Experiments

### 9.1 Recognition of Eight Simple Interactions

We have recognized the following eight simple (though composite) two-person interactions through our system: *approach, depart, point, shake-hands, hug, punch, kick,* and *push*. Interaction videos taken by Sony VX-2000 were converted into sequences of image frames with 320 × 240 pixel resolution, obtained at a rate of 15 frames per sec. Eight persons participated in the experiment and 24 sequences were obtained. In each sequence, participants were asked to perform a number of the above interactions consecutively and continuously. Overall, each simple interaction was performed 12 times.

```
LiquidConstCheck(Interaction i, Ranges r)
{
  G = Directed graph representation of i;
  for (all node n in G) {
    Initialize G[n]'s (min, max) with r;
  }
  for (all node s without a parent node) {
    result = AssignMinValue(s, G, 0);
    if(result==false) return false;
  }
  return true;
}

AssignMinValue(Node n, Graph G, int m)
{
  m++;
  if (G[n]'s min > m) m = G[n]'s min;
  else G[n]'s min = m;

  if(G[n]'s min > G[n]'s max) return false;
  else {
    if(visited[any parent of n]==false)
      return true;
    else {
      visited[s] = true;
      for (all child node c of n) {
        result = AssignMinValue(c, G, m);
        if (result==false) return false;
      }
    }
    return true;
  }
}
```

**Fig. 12** The algorithm for checking the temporal constraint of time range associated for an interaction

The representations for the eight interactions were constructed manually using our CFG-based representation scheme. Usually, a composite action is first defined in order to represent a meaningful one-person movement in the interaction. For example, in the previous sections, the composite action 'shake-hands action' was defined first in order to represent interaction 'shake-hands interaction'. The composite action 'shake-hands action' and the interaction 'touching' were sub-events.

Figures 13 and 14 show the intermediate outputs of each layer. In this experiment, two persons performed three interactions consecutively: shake-hands, point, and hug. The body-part layer extracts features for each body part per frame. Figure 13 shows the sequences of raw images, and processed images for extracting body-part parameters. Once the features for each frame are extracted, the pose layer con-

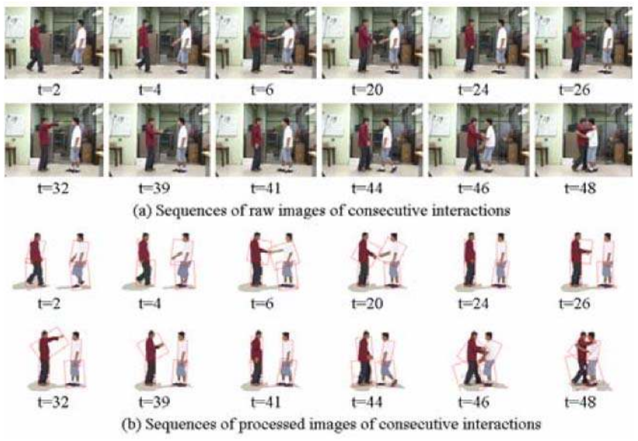
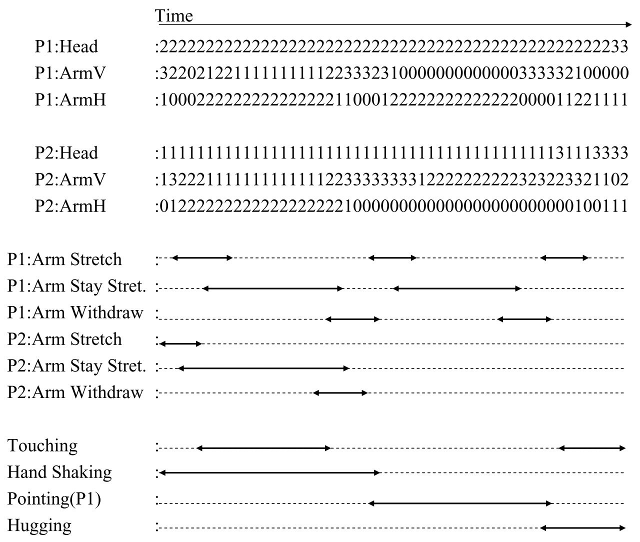

t=2
t=4
t=6
t=20
t=24
t=26
t=32
t=39
t=41
t=44
t=46
t=48
(a) Sequences of raw images of consecutive interactions
(b) Sequences of processed images of consecutive interactions

**Fig. 13** (**a**) Shows sequences of raw images of consecutive three interactions: shake hands, point, and hug. (**b**) Illustrates processed sequence of images by the body-part layer


```
P1:Head    :22222222222222222222222222222222222222222222222222233
P1:ArmV    :32202122111111111122333231000000000000003333321000000
P1:ArmH    :10002222222222222222110001222222222222222000011221111

P2:Head    :11111111111111111111111111111111111111111111131113333
P2:ArmV    :13222111111111111122333333331222222222223232233211102
P2:ArmH    :01222222222222222222100000000000000000000000000100111
```

**Fig. 14** Outputs of the pose layer, the gesture layer, and the action and interaction layer. Time intervals of atomic actions and interactions are presented

verts them into a discrete pose for each body part. The gesture layer converts sequences of poses into sequences of gestures. The recognition algorithms provided in the previous sections are then used to recognize interactions based on information from the gesture layer. Figure 14 shows the result of the pose layer, the gesture layer, and the final result of interaction recognitions.

Table 2 shows the performance (rate of true positives) of our recognition system on eight simple interactions. Because of the accurate representation and probabilistic recognition of composite actions, our system performed superior to the previous statistical system developed by Park and Aggarwal (2004b) that showed 0.8 recognition accuracy on the same activities. Moreover, the results are obtained from sequences of consecutive interactions, not segmented manually. The system was able to recognize sequences of actions and interactions with a high degree of accuracy. Our system taking advantage of hallucinations and the probabilistic recognition algorithm is compared with the deterministic version of the system. False positive rates of both systems were almost 0, since the probability of detecting multiple sub-events satisfying a particular temporal order 'by mistake' is extremely small.

**Table 2** A table showing recognition accuracy of eight simple interactions

| Interaction | Deterministic system | Probabilistic system |
|---|---|---|
| Approach | 12/12 | 12/12 |
| Depart | 12/12 | 12/12 |
| Point | 11/12 | 12/12 |
| Shake hands | 11/12 | 11/12 |
| Hug | 10/12 | 10/12 |
| Punch | 11/12 | 11/12 |
| Kick | 10/12 | 11/12 |
| Push | 11/12 | 11/12 |
| Total | 88/96 | 90/96 |

## 9.2 Recognition of High-Level Interactions

Most of the eight simple interactions presented above were constructed as a concatenation of composite actions and simpler interactions, such as *touching*. Those composite actions were constructed based on atomic actions. Therefore, a simple interaction can be interpreted as a composition of composite human actions (i.e. 3-levels). Our system has the ability to represent and recognize even higher-level interactions composed of those eight simple interactions. We conducted experiments on the recognition of two recursive interactions, *fighting* and *greeting*, which have a set of eight simple interactions as their sub-events. In addition, the high-level interaction *assault* and *pursuit* were also represented and recognized. The *assault* and *pursuit* contain the interaction *fighting* as their sub-event.

Our representation of the *fighting* and *greeting*, constructed using our a CFG-based representation scheme, are presented in Sect. 7. Both of them are represented recursively. The interaction *assault* and *pursuit* are also represented using CFG-based representation scheme, as a concatenation of the *fighting* interaction and other simpler interactions. The *assault* indicates a situation where one person approaches and tries to damage the other person who was staying peacefully. The interaction *fighting* follows as a consequence of the *assault*. The *pursuit* is initiated by two persons running in the same direction. If one person starts attacking the other and *fighting* follows as a consequence,

**Table 3** A table showing recognition accuracy of recursive interactions *fighting* and *greeting*

| Interaction | Deterministic system | Probabilistic system |
|---|---|---|
| Fighting | 13/18 | 17/18 |
| Greeting | 4/6 | 5/6 |
| Total | 17/24 | 22/24 |

**Table 4** A table showing recognition accuracy of recursive interactions *assault* and *pursuit*

| Interaction | Deterministic system | Probabilistic system |
|---|---|---|
| Assault | 4/6 | 6/6 |
| Pursuit | 4/6 | 6/6 |
| Total | 8/12 | 12/12 |

we call it *pursuit*. The actual representation of *assault* and *pursuit* is presented below.

```
AssaultInteraction($i, j$) = (
    list(  def($a$, Approach($i, j$)),
           list(  def($n$, NegativeInteraction($i, j$)),
                  def($f$, FightingInteraction($i, j$)))),
    and(   and(   meets($a, n$), meets($n, f$)),
           and(   starts($a$, this), finishes($f$, this)))
);
PursuitInteraction($i, j$) = (
    list(  def($m1$, Move($i$, direction1)),
           list(  def($m2$, Move($i$, direction1)),
                  list(  def($n$, NegativeInteraction($i, j$)),
                         def($f$, FightingInteraction($i, j$))))),
    and(   and(   meets($m1, n$), meets($m2, n$)),
           and(   meets($n, f$),
                  and(   starts($m1$, this), finishes($f$, this))))
);
```

Participants were asked to perform pure *fighting* and *greeting* interactions six times each. Also, *assault* and *pursuit* were performed six times. Since each *assault* or *pursuit* interaction contains one *fighting* interaction, the interaction *fighting* was performed 18 times total. Tables 3 and 4 show the recognition results of our system. Again, false positive rates were almost 0 because of the complex structure of the activities. In the case of recursive interactions such as *fighting* or *greeting*, multiple time intervals are detected for one long sequence of *fighting*. For example, if three consecutive *punching* actions occurred, then *fighting* interactions composed of one, two, and three *punching* actions must be detected. In this case, there are 3 *fighting* interactions composed of only one *punching*, 2 *fighting* interactions composed of two *punching*, and 1 *fighting* interaction composed of three *punching*. Only when the longest of them is detected, we evaluate that sequence as 'correct'. Figures 15 and 16 illustrate examples of time intervals detected for *fighting* and *assault*. We can observe that multiple time intervals are detected for each *fighting* interaction and *assault* interaction.

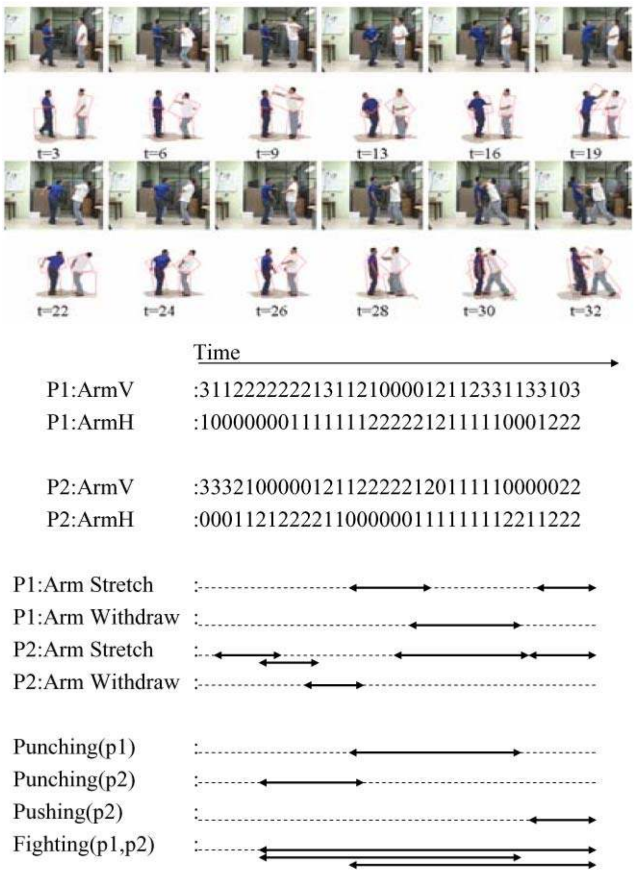

t=3
t=6
t=9
t=13
t=16
t=19
t=22
t=24
t=26
t=28
t=30
t=32
Time
P1:ArmV :31122222221311210000121123311 33103
P1:ArmH :100000001111111222221211111 0001222
P2:ArmV :3332100000121122222120111110000022
P2:ArmH :0001121222211000000111111112211222
P1:Arm Stretch
P1:Arm Withdraw
P2:Arm Stretch
P2:Arm Withdraw
Punching(p1)
Punching(p2)
Pushing(p2)
Fighting(p1,p2)

**Fig. 15** Time intervals of atomic actions, simple interactions, and the recursive interaction fighting are detected

## 9.3 Recognition with Noise

We have further conducted experiments to verify the applicability of our methodology on real-world systems. In the case of real-world applications such as an automated surveillance system using CCTV cameras, low-level components of the system including pose estimation and gesture detection tend to have relatively low accuracy due to noise and occlusions. What we verify through the experiment in this subsection is that the semantic layer we have proposed in this paper is able to reliably recognize high-level activities even under noisy low-level detections by probabilistically compensating for their failures. The motivation behind the design of the probabilistic semantic layer is to enable the system to handle noisy low-levels, and we here verify that our approach successfully handles them.

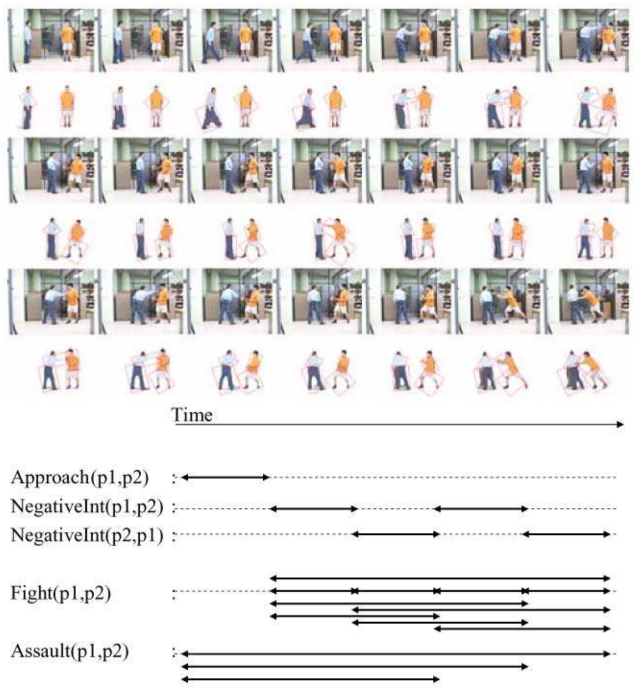


**Fig. 16** Example recognition results of the interaction assault

In order to simulate realistic environments where the low-level components (i.e. gesture recognition) have difficulties, we have added artificial noise to the gesture detection. The gesture detection accuracy of our low-level of the system is originally between 0.9 and 0.95 for datasets we have seen in Sects. 9.1 and 9.2. By making the system to randomly throwing out detected gestures with 0.2 probability, we were able to simulate the situation where the gesture detection accuracy is between 0.7 and 0.8. We have re-trained the parameters of our system for new noisy inputs, while setting the false positive rate to be approximately 0. The evaluations were done using the identical datasets that we have used in the previous subsections.

Table 5 shows the overall performance of our system. The performance of the deterministic version of our system and the probabilistic version of our system are measured and compared. Both systems are evaluated with normal gesture detection results as well as noisy gesture detection results.

As expected, the deterministic version of our system (i.e. the system without the probabilistic hallucinations) performed poorly on both the recognition of 8 simple interactions and that of recursive activities. Since all gestures composing an activity must be detected in order for the deterministic system to recognize the activity, the system missed many occurring activities. In contrast, the probabilistic version of our system was able to cope with noisy inputs. Especially in the case of activities composed of many gestures (e.g. pushing, shake-hands, fighting, assault, ...), the system was able to overcome few failures of the gesture detection based on the other gestures which are correctly detected. If the majority of gestures composing an activity were detected, our system was able to recognize the activity by generating probabilistic hallucinations for missing gestures.

**Table 5** A table comparing recognition accuracies of systems with noisy gesture detections

| System | 8-Simple | Recursive |
|---|---|---|
| Deterministic | 0.917 | 0.694 |
| Deterministic (noisy inputs) | 0.615 | 0.361 |
| Probabilistic | 0.938 | 0.944 |
| Probabilistic (noisy inputs) | 0.740 | 0.833 |

The total time complexity of our semantic layer on recognizing an activity is $O(\sum_{k=0\ to\ h}\ {}_nC_h \cdot r \cdot T)$ where $n$ is the number of the sub-events, $h$ is the maximum number of hallucinations allowed by the system, $r$ is the number of relationships of the activity, and $T$ is the number of frames. Generally, the maximum possible hallucinations, $h$, is set to be a small number, resulting overall complexity to be in a polynomial number. As a result, our semantic layer recognizes activities from gesture detection results in real-time with our semi-optimized C++ implementation. That is, if the low-level components of the system are functioning in real-time, the whole system is able to recognize activities in real-time from raw video sequences. In addition, what we must note is that the computational complexity of detecting an activity at each frame is dependent only on the number of sub-events $n$ and that of relationships described in its representation $r$. This implies that the complexity is not dependent on the total number of possible gestures, making our system scalable.

## 10 Conclusions and Future Works

We have presented the general methodology for the automated recognition of complex human activities. The fundamental idea is to use the CFG-based representation scheme to represent high-level actions and interactions. The CFG-based representation scheme provides a formal method to define occurring time intervals of composite actions and interactions. The idea of hierarchically representing complex actions and interactions as a composition of simpler actions and interactions was the key. Based on the constructed representation of activities, the recognition was performed hierarchically and probabilistically. Our experiments show that the system can represent and recognize complex human activities with a high recognition rate.

The novelty and contribution of our work is in the framework and algorithms to represent and recognize high-level hierarchical activities from a raw image sequence. The algorithm enables the recognition of human activities with any level of hierarchy, whose sub-events are organized sequentially and/or concurrently. In addition, our representation explicitly captures the hierarchical and recursive nature of actions and interactions. Recursive representation was allowed to describe high-level activities, enabling the system to recognize human activities with a continuous characteristic. Our system has the ability to use represented actions as sub-events of higher-level actions, thereby minimizing the redundancy. Finally, the recognition process was probabilistic, enabling the system to handle noisy inputs and compensate for the failures of low-level processing. The main drawback of our system is in the time complexity to find optimum recognition solution: it is a NP-hard problem. Our system overcomes this by applying heuristics and posing constraints on the recognition process. Experimental results suggest that our methodology is reasonable and gives good results.

The potential of our work is that our methodologies are applicable to various high-level activity recognition problems, if domain dependent atomic-level actions can be detected with its starting time and ending time using other computer vision techniques. Our system can recognize any action and interaction if their time intervals can be defined properly through our CFG-based representation scheme. The representation and recognition methodologies in our semantic layer are robust and reliable, even compensating for the complete failures of mistakes made by low-level component like HMMs. In the future, we plan to apply this framework to various applications, including surveillance systems using CCTVs, human-computer interactions systems, and a system for sports play analysis.

Also, we aim to develop a methodology that is able to learn representations of activities automatically based on large training sets. Currently, the representation of human activities are encoded by human experts following our CFG syntax. On average, once an expert gets familiar with our CFG representation syntax, he/she is able to encode a representation of a single activity in few minutes. Our representation essentially is equivalent to the ordering of time intervals (e.g. Fig. 5), and converting it into our formal representation is a mechanical process if the expert is confident about its conceptual structure. However, even though the representation of one activity can be used repeatedly for its recognition once encoded, adding the representation whenever a new activity is required to be recognized is an inefficient task.

Therefore, in our future work, we aim to enable automated learning of the representation of human activities. Whenever an activity is occurring, it is likely to show particular sub-events organized in a particular order. Thus, grouping commonly appearing gestures and capturing common structural patterns among them will enable the construction of appropriate representations. The learned representation is expected to have several variations depending on the training set, similar to the case of experts with different bias encoding the representation. However, what we have observed though our experiments is that even though the representations of intermediate actions created by different experts vary, the representations encoded are eventually decomposing the activity into a similar set of gestures. Since the probabilistic recognition process of our system is mostly dependent on the gestures composing the activity, the recognition with automatically learned representation is expected to show the comparable performance as long as it is able to capture them correctly.

## Appendix A

In this appendix, we present full context-free grammar (CFG) syntax of our activity representation. We first provide a brief explanation of general context-free grammars. Next, the full syntax for our representation is described.

### A.1 Context-Free Grammar

In the formal language theory, context-free grammars denote a particular class of languages which generate strings of terminals (i.e. symbols). CFG has first been defined by Chomsky (1956), and it has often been used to describe syntax of programming languages, such as *Algol*, C, C++, and *Java*. The advantage of CFGs is that they are expressive enough to describe complex languages like programming languages, while parsing them is computationally tractable. Efficient polynomial time parsers (e.g. LL and LR parsers) have been developed.

A context-free grammar is described based on four components. We represent a CFG $G$ by four components, as $G = (V, T, P, S)$. $V$ is a set of variables, $T$ denotes terminals, $P$ describes a set of production rules, and $S$ is the starting variable. Each variable, also called a *non-terminal*, corresponds to a set of strings that can be generated by following production rules starting from the variable. Terminals are a finite set of symbols that appear in a language. For example, integer numbers are terminals for a numerical expression. Production rules are the rules to generate a sequence of terminals. By consecutively applying productions rules to replace variables with a sequence of terminals, a string of symbols following the syntax can be constructed. The start symbol is the variable which the application of production rules has to be started with.

Let's look into an example CFG for generating mathematical expressions. For the simplification, we limit our expressions to the operators $+$ and $*$, which represent the addition and the multiplication. We also limit the expressions to use only $a$, $b$, and $c$ as their symbols. The full syntax

$G = (V, T, P, S)$ can be expressed as follows: $V = \{E, I\}$, $T = \{a, b, c\}$, $S = E$, and $P$ which is listed below.

$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow I$

$I \rightarrow a$

$I \rightarrow b$

$I \rightarrow c$

As a result, strings such as $a + b$, $a * b + a$, and $a + a + b + c * b$ can be generated. Only the strings that can be generated by the production rules are the valid strings satisfying the syntax. Thus, strings such as $-a$, $ab+$, and $d * e$ are invalid strings which are syntactically incorrect.

### A.2 Full CFG Representation Syntax

In this subsection, we present the full CFG representation syntax for human activities. We describe the representation syntax for atomic actions, composite actions, and interactions. We start with the presentation of the syntax for actions (i.e. atomic actions and composite actions), and then describe the syntax for interactions.

The CFG syntax presented in this paper is a detailed and formalized version of the CFG syntax presented in Ryoo and Aggarwal's earlier works (Ryoo and Aggarwal 2006a, 2006b). As mentioned in the previous subsection, a CFG $G$ consists of four components: $G = (V, T, P, S)$ where $V$ is the set of variables, $T$ is the terminals, $P$ is the set of productions, and $S$ is the starting variable. In the case of our CFG syntax for actions, $V = \{$ ActionDefine, ActionName, Action, ActionExp, ActionDefs, ActionRelationship, LogicalPredicate, Temporal-Predicate, name, $c$, var, person, operation triplet $\}$, $T = \{$ atomic_ action, list, def, null, 'this', and, or, not, before, meets, overlaps, starts, during, finishes, (, ), =, ;, ,, {all alphabet characters} $\}$, and $S =$ ActionDefine. $P$, the production rules, is presented below. Parameters were added to the variables when describing the production rules $P$, to illustrate the semantic constraints for composite actions. Our $P$ is a formal and complete description of production rules for our representation syntax without any parameters. Parameters were added not to describe syntax, but to describe semantic characteristics of our representation.

Non-terminal ActionExp($i$) indicates the action of person $i$. ActionExp($i$) can be either an atomic action, or a composite action defined with two components: ActionDefs($i$, var) and ActionRelationship(var). The first component, ActionDefs($i$, var), defines the variables for corresponding time intervals of sub-events. Parameter var is defined to be the list of variables associated with sub-events. ActionDefs($i$,var) is the list of several def($c$, Action($i$)), and this defines the contents of list var. Statement def($c$, Action($i$)) associates a variable with the time interval of a denoted sub-event. As a result, list var contains a list of variables associated with time intervals of the corresponding sub-events.

The second component is ActionRelationship(var). With temporal and logical predicates, ActionRelationship(var) defines all necessary conditions for the action using all variables in var and the special variable 'this'. A combination of any temporal predicates presented in Sect. 5.2 can be used to define ActionRelationship(var). The time interval 'this' satisfying all necessary conditions will be the corresponding time interval for the action.

ActionDefine($i$)
  $\rightarrow$ ActionName($i$) "=" ActionExp($i$)";"
ActionName($i$) $\rightarrow$ name"("person($i$)")"

Action($i$) $\rightarrow$ ActionExp($i$) | ActionName($i$)
ActionExp($i$)
  $\rightarrow$ "("ActionDefs($i$, var)","
      ActionRelationship(var)")"
  | "atomic_action" "("operation triplet")"

ActionDefs($i$, var)
  $\rightarrow$ "list" "(" "def" "("$c$", "Action($i$)")" ","
      ActionDefs($i$, var $- c$)")"
  | "def" "("$c$", "Action($i$)")"
  | "null"
ActionRelationship(var)
  $\rightarrow$ LogicalPredicate"("ActionRelationship(var)","
      ActionRelationship(var)")"
  | TemporalPredicate"(" "this" "," "var($a$)")"
  | TemporalPredicate"("var($a$)"," "this" ")"
  | TemporalPredicate"("var($a$)", "var($b$)")"
  | "null"

LogicalPredicate $\rightarrow$ "and"|"or"|"not"
TemporalPredicate $\rightarrow$ "before"|"meets"|"overlaps"
  |"starts"|"during"|"finishes"
name $\rightarrow$ char$*$
$c$ $\rightarrow$ char$*$
var $\rightarrow$ char$*$
person $\rightarrow$ char$*$

Similarly, the production rules for the interaction representation can be described as follows. The only difference between the production rules for composite actions and that for interactions is that the production rules for interactions allow description of two different persons as participants. As a result, the spatial predicates describing spatial relations between two persons may also be listed following the syntax.

```
InteractionDefine(i, j)
  →   InteractionName(i, j)“ = ”InteractionExp(i, j)“; ”
InteractionName(i, j)
  →   name“(”person(i)“, ”person(j)“)”

Interaction(i, j)
  →   InteractionExp(i, j)
  |   InteractionName(i, j)
InteractionExp(i, j)
  →   “(”InteractionDefs(i, j, var)“, ”
                    InteractionRelationship(i, j, var)“)”

InteractionDefs(i, j, var)
  →   “list” “(” “def” “(”c“, ”Interaction(i, j)“)” “, ”
                    InteractionDefs(i, j, var − c)“)”
  |   “list” “(” “def” “(”c“, ”Action(i or j)“)” “, ”
                    InteractionDefs(i, j, var − c)“)”
  |   “def” “(”c“, ”Interaction(i, j)“)”
  |   “def” “(”c“, ”Action(i or j)“)”
  |   “null”
InteractionRelationship(var)
  →   LogicalPredicate“(”
                    InteractionRelationship(var)“, ”
                    InteractionRelationship(var)“)”
  |   TemporalPredicate“(” “this” “, ”var(a)“)”
  |   TemporalPredicate“(”var(a)“, ” “this” “)”
  |   TemporalPredicate“(”var(a)“, ”var(b)“)”
  |   SpatialPredicate“(”person(i)“, ”person(j)“, ”
                    threshold“)”
  |   “null”

SpatialPredicate        → “near”|“touch”
```

## References

Allen, J. F., & Ferguson, G. (1994). Actions and events in interval temporal logic. *Journal of Logic and Computation*, *4*(5), 531–579.

Bobick, A. F., & Wilson, A. D. (1997). A state-based approach to the representation and recognition of gesture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(12), 1325–1337.

Chomsky, N. (1956). Three models for the description of language. *IEEE Transactions on Information Theory*, *2*(3), 113–124.

Fine, S., Singer, Y., & Tishby, N. (1998). The hierarchical hidden Markov model: Analysis and applications. *Machine Learning*, *32*(1), 41–62.

Hongeng, S., Nevatia, R., & Bremond, F. (2004). Video-based event recognition: Activity representation and probabilistic recognition methods. *Computer Vision and Image Understanding: CVIU*, *96*(2), 129–162.

Ivanov, Y. A., & Bobick, A. F. (2000). Recognition of visual activities and interactions by stochastic parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(8), 852–872.

Joo, S.-W., & Chellappa, R. (2006). Attribute grammar-based event recognition and anomaly detection. In *CVPRW ’06: Proceedings of the 2006 conference on computer vision and pattern recognition workshop* (p. 107).

Minnen, D., Essa, I. A., & Starner, T. (2003). Expectation grammars: Leveraging high-level expectations for activity recognition. In *CVPR(2)* (pp. 626–632). IEEE Computer Society.

Moore, D. J., & Essa, I. A. (2002). Recognizing multitasked activities from video using stochastic context-free grammar. In *AAAI/IAAI* (pp. 770–776).

Natarajan, P., & Nevatia, R. (2007). Coupled hidden semi Markov models for activity recognition. In *IEEE workshop on motion and video computing, 2007. WMVC ’07* (pp. 10–10).

Nevatia, R., Hobbs, J., & Bolles, B. (2004). An ontology for video event representation. In *CVPRW ’04: Proceedings of the 2004 conference on computer vision and pattern recognition workshop (CVPRW’04)* (Vol. 7, p. 119).

Nguyen, N. T., Phung, D. Q., Venkatesh, S., & Bui, H. H. (2005). Learning and detecting activities from movement trajectories using the hierarchical hidden Markov models. In *CVPR(2)* (pp. 955–960). IEEE Computer Society.

Oliver, N. M., Rosario, B., & Pentland, A. P. (2000). A Bayesian computer vision system for modeling human interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(8), 831–843.

Park, S., & Aggarwal, J. K. (2004a). A hierarchical Bayesian network for event recognition of human actions and interactions. *Multimedia Systems*, *10*(2), 164–179.

Park, S., & Aggarwal, J. K. (2004b). Semantic-level understanding of human actions and interactions using event hierarchy. In *CVPRW ’04: Proceedings of the 2004 conference on computer vision and pattern recognition workshop (CVPRW’04)* (Vol. 1, p. 12).

Park, S., & Aggarwal, J. K. (2006). Simultaneous tracking of multiple body parts of interacting persons. *Computer Vision and Image Understanding*, *102*(1), 1–21.

Pinhanez, C. (1999). *Representation and recognition of action in interactive spaces*. MIT Media Lab, June 1999: PhD thesis.

Ryoo, M. S., & Aggarwal, J. K. (2006a). Recognition of composite human activities through context-free grammar based representation. In *CVPR ’06: Proceedings of the 2006 IEEE computer society conference on computer vision and pattern recognition* (pp. 1709–1718).

Ryoo, M. S., & Aggarwal, J. K. (2006b). Semantic understanding of continued and recursive human activities. In *ICPR ’06: Proceedings of the 18th international conference on pattern recognition* (pp. 379–382).

Ryoo, M. S., & Aggarwal, J. K. (2007). Robust human-computer interaction system guiding a user by providing feedback. In M. M. Veloso (Ed.), *IJCAI 2007, Proceedings of the 20th international joint conference on artificial intelligence* (pp. 2850–2855).

Shi, Y., Huang, Y., Minnen, D., Bobick, A. F., & Essa, I. A. (2004). Propagation networks for recognition of partially ordered sequential action. In *CVPR(2)* (pp. 862–869).

Siskind, J. M. (2001). Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic. *Journal of Artificial Intelligence Research (JAIR)*, *15*, 31–90.

Starner, T., & Pentland, A. (1995). Real-time American sign language recognition from video using hidden Markov models. *ISCV*, *00*, 265.

Vu, V.-T., Brémond, F., & Thonnat, M. (2003). Automatic video interpretation: A novel algorithm for temporal scenario recognition. In G. Gottlob & T. Walsh (Eds.), *IJCAI-03, Proceedings of the eighteenth international joint conference on artificial intelligence* (pp. 1295–1302). San Mateo: Morgan Kaufmann.

Yamato, J., Ohya, J., & Ishii, K. (1992). Recognizing human action in time-sequential images using hidden Markov model. In *CVPR* (pp. 379–385).