Semantic Understanding of Continued and Recursive Human Activities

M. S. Ryoo and J. K. Aggarwal
Computer and Vision Research Center, Department of ECE
The University of Texas at Austin, Austin, TX 78712
{mryoo, aggarwaljk}@mail.utexas.edu

Abstract

This paper presents a methodology for semantic understanding of complex and continued human activities. A context-free grammar (CFG) based representation scheme developed earlier is extended to construct a description for continued and recursive human activities. New system recognizes recursively described high-level interactions: fighting and greeting. The system understands activities by detecting the time intervals that satisfy their semantic descriptions.

1. Introduction

Semantic understanding of human activity is an important component in several computer vision applications. Particularly, for surveillance systems, understanding what activity people are performing is needed, in order to detect abnormal activities as opposed to the normal activity of people using public places like airports and subway stations. Semantic understanding of activities is also essential for real-time automatic monitoring of elderly people, patients, or babies. Several researchers have achieved semantic understanding of human activities. However, most of those works focus mainly on the understanding of single (i.e. atomic) actions of humans, not on the understanding of the complex composition of multiple movements or actions [1].

Our system understands continued and recursive human activities semantically through the use of a context-free grammar (CFG) based representation scheme. Our work is an extension of previous work done by Ryoo and Aggarwal [6]. Their representation scheme enables a user to describe composite human activities based on simpler sub-activities. The significant extension made in this paper is that our system allows recursive descriptions of composite activities, which is the key for understanding high-level human activities such as fighting and greeting. Recursive description means that a user is able to describe high-level activities as a composition of other simpler activities and the defining activity itself.

The focus of this paper is on the semantic layer, the highest level of the recognition system. In the semantic layer, recursive descriptions of human activities are constructed in terms of sub-events' time intervals and the relationships among them. The system then understands human activities by finding a time interval that satisfies all conditions specified in the description. Our system classifies human activities into three categories: atomic actions, composite actions, and interactions [6]. A hierarchical framework developed by Park and Aggarwal [4,5] is adopted to detect atomic actions from sequence of input frames.

2. Previous systems

Researchers in traditional AI fields have focused on definitions and inferences of events. Allen and Ferguson [2] presented a definition of temporal intervals, and defined events using interval temporal logic. Descriptions for events were constructed in the form of the first-order logic, specifying necessary relationships for the events.

Several researchers in computer vision conducted research on formal description of human activities, motivated by the Allen's interval temporal logic. The representation language for composite events presented by Nevatia *et al.* [3] yielded successful results. They not only constructed the formal description, but also illustrated algorithm to detect ongoing human activities from input images. The limitation of their work was on describing complex compositions of activities which are composite activities themselves, since the structure of the descriptions was strictly fixed to three layers.

Ryoo and Aggarwal [6] also adopted the concept of time interval representation presented by Allen and

Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06)
0-7695-2521-0/06 $20.00 © 2006 IEEE

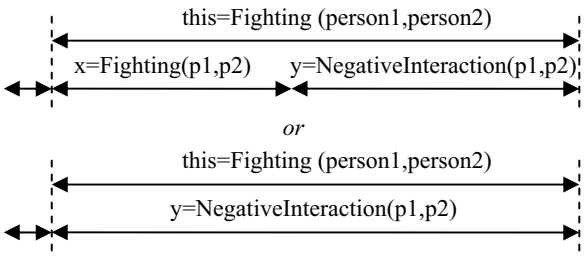

Figure 1: The necessary temporal relationship among time intervals for recursive interaction *Fighting*. The bottom figure shows the base case of the *Fighting*.

Ferguson to describe hierarchical events. They used the context-free grammar as a format of the representation scheme. Their scheme enabled the user to describe higher-level activities based on already described simpler sub-events, allowing the system to handle infinite layers of hierarchy. The detection of described activities is essentially the traditional constraint satisfaction problem.

3. Recursive activities

Previous researchers described human activities in terms of a strictly fixed number of sub-events. However, for complex human activities such as fighting or greeting, the number of sub-events is not clear. We cannot say that one person punching the other three times is a fighting activity while the person punching four times is not. Instead, such high-level human activities tend to have recursive characteristics. Assume that the system detected 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 activity, based on the detection of the shorter fighting interaction.

3.1 Recursive activity description

We here provide a concept of recursive description for human activities. In the case of recursive actions and interactions, a defining human activity can become a sub-event of itself. We do not describe the actual overall size of the defining human activity, but instead use a shorter identical activity as a sub-event. This recursive description is able to catch infinite size of the defining activity, since the level of hierarchy can grow infinitely.

Essentially, the syntax provided in the previous 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 could only denote activity names (strings of characters) that had been defined already. 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 described easily. The formal production rules of the CFG representation scheme are outlined below.

InteractionDefine(i, j)
-> InteractionName(i, j) = InteractionExp(i, j);
Interaction(i, j)
-> InteractionExp(i, j) | InteractionName(i, j)
InteractionsExp(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))
InteractionRelationship(i, j, var)
-> Logical-Predicate(InteractionRelationship(i,j, var),
InteractionRelationship(i, j, var))
| Temporal-Predicate('this', var.element)
| Temporal-Predicate(var.element, var.element)
| Spatial-Predicate(person i, person j, threshold)

Another important structure of the recursive activities is the existence of the base case. Recursive activities are detected by detecting the shorter identical activity. Therefore, at some point, the system needs the seed (or core) activity for detection, which does not rely on the detection of the shorter identical activity. This is called base case of the recursive activity. When describing 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 another shorter *Fighting* and one *NegativeInteraction* as illustrated in Figure 1. The *NegativeInteraction* is an *or* concatenation of *Punching, Kicking,* and *Pushing*, which compose activities of the *Fighting* sequence. The base case of the *Fighting* is one *NegativeInteraction*. The *Greeting* interaction can be described in a similar manner. The *Greeting* 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 *Shake-hands* interaction. The formal description for *Negative-Interaction, Fighting,* and *Greeting* is as follows:

```
NegativeInteraction(i, j) =  (
    list(   def('x', Punching (i, j)),
            list(  def('y', Kicking (i, j),
                   def('z', Pushing (i, j)) )),
    or(     equals('this', 'x'),
            or(  equals('this', 'y'), equals('this', 'z') ))
);
Fighting(i, j) = (
    list(   def('x', Fighting(i, j)),
            def('y', NegativeInteraction(i, j)) ),
    or( equals('y', 'this'),
        and(meets('x', 'y'),
            and(starts('x', 'this'), finishes('y', 'this'))))
);
Greeting(i, j) = (
    list(   def('x', Shake-hands(i, j)),
            list(  def('y', PositiveInteraction(i, j))
                   def('z', Greeting(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') ) ) )
    )
);
```

3.2 Recursive activity understanding

For the understanding of recursively described actions and interactions, an iterative approach is used. We continue to explain this iterative algorithm with the example of the *Fighting* interaction. The system starts with setting time interval 'x', corresponding to the sub-event *Fighting*, to null. Then, the system is only able to find base cases. In the case of *Fighting*, single *NegativeInteraction* corresponds to a base case. Once we find some initial *Fighting* interactions, we treat them 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 further *Fighting* is detected. The detection result of n^{th} iteration is served as a sub-event of $(n+1)^{th}$ iteration. With this iterative algorithm, given the detection result of 1st iteration, i.e. base case detection, we are able to detect recursive activities. The pseudo-code of the complete algorithm is provided in Figure 2. The function CSP implies that we are solving constraint satisfaction problem mentioned in our previous work [6]. The system finds occurring activity by checking all possible combinations of (variable, time interval) pairs, whether they satisfy the activity's description 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;
  }
  list.x = null;
  while (true) {
     result = CSP(list, i);
     if(list.x==result) break;
     else list.x = result;
  }
  return result;
}
```

Figure 2: The algorithm for understanding of recursive activities.

4. Experiments

The goal of our system is to understand the recursive interactions, *Fighting* and *Greeting*. Also, the following eight simple interactions are described and detected as sub-events of *Fighting* and *Greeting*: *Approach, Depart, Pointing, Shake-hands, Hugging, Punching, Kicking,* and *Pushing* [6]. The descriptions for these simple interactions and two recursive interactions were constructed manually following the CFG representation scheme. Usually, composite actions are first defined in order to describe meaningful one-person movement, based on several atomic actions. Eight simple interactions are then constructed, having composite actions as their sub-events. Those interactions serve as sub-events for higher-level recursive interactions.

Interaction videos produced using a Sony VX-2000 are converted into sequences of image frames with 320*240 pixel resolution obtained at a rate of 15 frames per sec. Six pairs of 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, and *Fighting* and *Greeting* were performed 6 times each throughout all sequences.

Figure 3 presents an understanding of the process of the complex interaction *Fighting*. Person2 *Punching*, person1 *Punching*, and person2 *Pushing* occurred sequentially. Poses for the body part 'arm' are illustrated along the time line. In our system, smaller

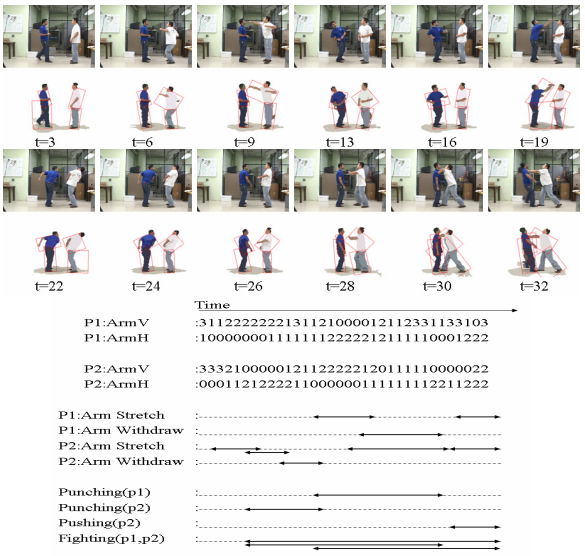

Figure 3: Time intervals of atomic actions, simple interactions, and recursive interaction *Fighting* are detected.

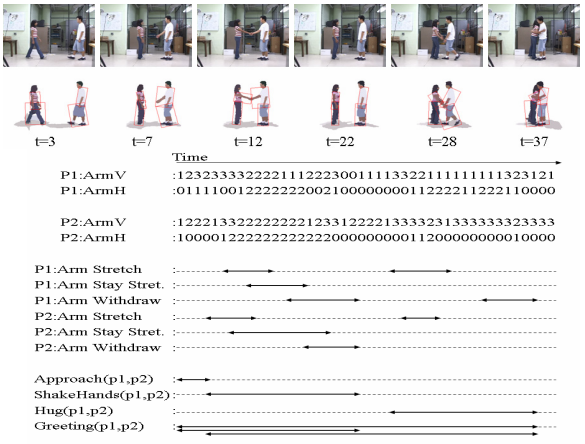

Figure 4: Recursive interaction *Greeting*.

ArmV pose value implies a higher arm, and smaller ArmH implies a more withdrawn arm. The atomic action detection results are illustrated as time intervals. The system detects simple interactions based on detected atomic actions. Then, the system detects *Fighting* interaction with the recursive understanding algorithm we discussed. Figure 4 presents a recursive interaction *Greeting*.

Tables 1 and 2 show the performance of our system. Because of the accurate description of recursive activities, the system was able to detect activities with varying length. Moreover, results are obtained from sequences of consecutive interactions, not segmented manually. The system automatically understands what activity occurred where, without manual segmentation.

5. Conclusions

We have presented a semantic description scheme and understanding algorithm for continued and recursive human activities. The fundamental idea is to extend the previously developed CFG representation scheme to allow the recursive description of human activities. Methodology for the semantic understanding of recursive human activities is provided clearly. Our experiments show that the system understands continued and recursive human activities with a high recognition rate.

Table 1: Detection of recursive interactions

interaction	total	correct	accuracy
fighting	6	4	0.667
greeting	6	4	0.667
total	12	8	0.667

Table 2: Detection of simple interactions

interaction	total	correct	accuracy
approach	12	12	1.000
depart	12	12	1.000
pointing	12	11	0.917
shake hands	12	11	0.917
hugging	12	10	0.833
punching	12	11	0.917
kicking	12	10	0.833
pushing	12	11	0.917
total	96	88	0.917

References

[1] J.K. Aggarwal and Q. Cai, "Human Motion Analysis: A Review", *CVIU* 73(3), 1999, pp. 295-304.

[2] J.F. Allen and G. Ferguson, "Actions and Events in Interval Temporal Logic", *Journal of Logic and Computation* 4(5), 1994, pp. 531-579.

[3] R. Nevatia, T. Zhao, and S. Hongeng, "Hierarchical Language-based Representation of Events in Video Streams", *Proceedings of the Workshop on Event Mining*, 2003.

[4] S. Park and J. K. Aggarwal, "A Hierarchical Bayesian Network for Event Recognition of Human Actions and Interactions", *ACM Journal of Multimedia Systems,* special issue on Video Surveillance 10(2), 2004, pp. 164-179.

[5] S. Park and J. K. Aggarwal, "Simultaneous tracking of multiple body parts of interacting persons", *CVIU* 102(1), April 2006, pp. 1-21.

[6] M.S. Ryoo and J.K. Aggarwal, "Recognition of Composite Human Activities through Context-Free Grammar based Representation", *CVPR'06*, New York, NY, 2006

[7] J.M. Siskind, "Grounding the Lexical Semantics of Verbs in Visual Perception using Force Dynamics and Event Logic", *Journal of Artificial Intelligence Research* 15, 2001, pp. 31-90.