Recognition of High-level Group Activities Based on Activities of Individual Members

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

Abstract

The paper describes a methodology for the recognition of high-level group activities. Our system recognizes group activities including group actions, group-persons interactions, group-group interactions, inter-group interactions, and their combinations described using a common representation scheme. Our approach is to represent various types of complex group activities with a language-like representation, and then to recognize represented activities based on the recognition of activities of individual group members. A hierarchical recognition algorithm is designed for the recognition of high-level group activities. The system was tested to recognize activities such as 'two groups fighting', 'a group of thieves stealing an object from another group', and 'a group of policemen arresting a group of criminals (or a criminal)'. Videos downloaded from YouTube as well as videos that we have taken are tested. Experimental results shows that our system recognizes complicated group activities, and it does it more reliably and accurately compared to previous approaches.

1. Introduction

A significant amount of research has addressed the recognition of human activities recently. Researchers are particularly successful in recognizing the activities of one individual or between two individuals, such as pushing and fighting. Notably, we in our previous work [8] have presented a representation scheme to describe high-level human-human interactions based on their sub-events, and proposed a hierarchical algorithm to recognize represented interactions. Not only simple interactions such as punching, kicking, and shaking hands are recognized, but also recursive interactions like 'fighting' between two persons are recognized with our previous framework. In this paper, we take our next evolutionary step in human activity recognition: recognition of group activities.

Group activities are the activities that can be characterized by movements of members who belong to one or more conceptual groups. Recognition of groups and their activities will make detection of high-level events possible, which are semantically meaningful when overall actions of multiple persons are considered jointly but not when they are considered individually. Automated recognition of suspicious groups and their activities such as 'a group of thieves robbing the bank' are essential for the construction of high-level surveillance systems. The analysis of movements and plays in team sports also becomes possible with the group activity recognition system. The semantic understanding of military operations and joint works is another application of group activity recognition.

In this paper, we present a novel methodology for the recognition of high-level group activities. Our approach is to encode human knowledge on the structure of group activities, and make the system recognize group activities based on their representation hierarchically. That is, we are crossing the horizon of previous description-based human activity recognition approaches [5,10] toward the recognition of group activities. We believe that ours is the first paper presenting general recognition methodology for group activities with complex temporal, spatial, and logical structures. We focus on both a new format for the group activity representation and a new recognition algorithm.

Unlike previous group activity recognition systems, our system is designed to represent and recognize as broad range of high-level group activities as possible (including group actions, group-persons interactions, group-group interactions, and inter-group interactions), making the system more robust and generally applicable. Khan and Shah recognized group activities based on rigidity formation [6]. They focus on recognizing spatially structured groups and their activities such as a parade which is a type of 'all members of a group showing identical action'. Researchers have recognized inter-group fighting (a fight between two members of a group) which has been labeled as a group activity in PETS'04 dataset [7]. Zhang et al. [11] recognized inter-group interactions among limited number of participants in a meeting room using multi-layered hidden Markov models. Gong and

Xiang [3] successfully recognized interactions between multiple objects using dynamic probabilistic networks. Their system also focuses on only one class of group activity, inter-group interactions, and the number and types of participants are also fixed, as in [11]. Similarly, Cupillard *et al.* [2] recognized inter-group interactions with fixed number of participants. In addition, most of previous works assume that group members are spatially separable from non-members in order to recognize group activities.

Our system describes group activities in terms of a formal representation using context-free grammar (CFG) as its syntax. A group activity is decomposed into several single person actions and person-person interactions among members of groups, and our language-like representation describes the activity by attaching universal quantifiers ($\forall$) and/or existential quantifiers ($\exists$) to those sub-events. For example, the group activity 'all members are carrying their baggage' must be represented by applying the universal quantifier to the individual activity 'a person carries baggage', while 'one member of the group raises his/her hand' must be represented by applying the existential quantifier to 'a person raises his/her hand'. Spatial constraints such as 'all members of one group should be spatially close' must also be listed as well.

Our system recognizes group activities by searching for individuals whose activities satisfy the representation of the group activity. That is, our system does not rely on the correct segmentation of groups based on spatial information like most of previous systems. In our approach, individual activities of persons in the scene are first recognized, and then used for the group activity recognition by comparing them with the representation. For example, recognition of the group activity 'all members are carrying their baggage' is done by detecting individuals who performed the activity 'a person carries baggage' at the same time. As a result of the algorithm, group activities and groups performing the group activities are recognized simultaneously. Figure 1 shows example group activities.

2. Representation

The approach we take to represent high-level group activities is to decompose them into several simpler activities, which we call 'sub-events' of the activity. Sub-events of a group activity can be actions of a group member, interactions between the members, and/or other group activities of the same group. We first discuss different types of group activities that our system represents, and then present formal representation syntax.

2.1. Types of group activities

We categorize group activities by considering the number of participating groups, the number of participants not in any group, and types of the activities' sub-events.

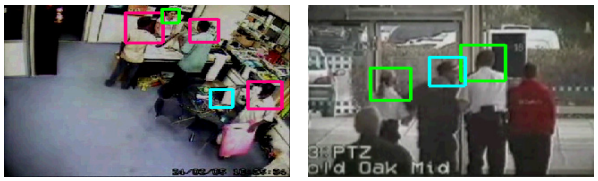

Figure 1: Snapshots of group activities. The left figure shows a group-group interaction, 'group stealing'. The right figure shows a group-group interaction, 'group arresting'.

Group actions. If a group activity can be specified only using actions of its group members, we call it a 'group action'. 'Marching' is a typical example of group actions: the activity can be characterized as all group members showing one type of individual action, 'moving'. The 'marching' can be denoted as *March(Group G1)*.

Group-persons interactions. If a group as well as persons outside of the group participates in the activity, we denote it as a 'group-persons interaction'. The activity 'march by signal', which indicates an activity where a group starts marching after getting an order from a commander outside the group, is an example. 'March by signal' is denoted in the form of *MarchBySignal(Group G1, Person p1)*.

Group-group interactions. Two groups fighting and two groups having a conversation are good examples of group-group interactions. A group-group interaction can be composed of the actions of a group member of any group and/or interactions between two members from each group. A group-group fighting can be notated as *GroupGroup-Fighting(Group G1, Group G2)*.

Inter-group interactions. 'Inter-group interactions' are group activities that involve interactions between members of a same group as sub-events. A group activity indicating that two members of a group are fighting is an example of inter-group interactions: *InterGroupFighting(Group G1)*.

Combinations. Our system is designed to represent group activities of above-mentioned four elementary types as well as more complicated activities that can be decomposed into several group activities of the elementary types (i.e. interactions between multiple groups and persons).

2.2. Group activity representation

We present general representation syntax that is able to describe group activities of any of above-mentioned categories hierarchically. The concept of the 'member variables' and the 'group spatial predicates', which have not been covered by any of previous activity representation methodologies, are newly introduced to denote participating group members and to describe spatial constraints needed among the group members. Based on new concepts and predicates, we represent a group activity in terms of time intervals of activities of individual

members (or other simpler group activities) composing it, the relationship specifying the temporal structure among sub-events, and necessary spatial conditions among group members. Detailed context-free grammar (CFG) syntax of our representation is presented.

Member variables. A member variable is a variable used to denote one arbitrary member or all members of a group. We attach an existential quantifier ($\exists$) or a universal quantifier ($\forall$) to a member variable, in order to describe conditions that have to be satisfied by one member or all members of a group. If an existential quantifier is attached to a member variable, there has to be at least one individual member of the group who can be associated with the member variable to make related conditions to be true. If a universal quantifier is attached, all members of the group must be able to be associated with the member variable. That is, by using member variables as participants of sub-events, we are able to describe sub-events need to be performed by all group members or by any one member. In addition, sub-events need to be performed by the same individual may also be specified by using the same member variable as their participant. Our syntax to represent a list of member variables and its example are presented below.

MemberVariableDefs -> MemberVariableDef "," MemberVariablsDefs
| MemberVariableDef
MemberVariableDef -> Quantifier person_var "in" group_var
Quantifier -> "∀" | "∃"
Ex> *∀ a in G1, ∃ b in G2, ∃ c in G3, ...*

Time intervals. A time interval specifies a starting time and an ending time of an occurring sub-event. A group activity is composed of multiple sub-events whose participants are specified using member variables and/or other non-member participants. In order to describe temporal structure of a group activity, both the sub-events composing the group activity and their time intervals must be listed. The formal syntax is as follows:

TimeIntervalDefs
-> "def" "(" time_var "," Activity ")"
| "list" "(" "def" "(" time_var "," Activity ")" "," TimeIntervalDefs ")"
Ex> *list(def(t1, Carrying(a)), def(t2, ...))*

Predicates. Predicates are binary functions that are used to describe temporal, spatial, and logical relationships needed for the activity. Our system adopts Allen's temporal predicates (*before, meets, overlaps, during, starts, finishes,* and *equals*) [1], which have been widely used to specify temporal structures. Spatial predicates between individual persons, *near* and *touch*, are also used. Spatial predicates for describing the spatial status of a group are newly designed and added for the representation, whose definition is listed below. The predicate *dense* and *sparse* describe whether all group members are close to each other or not. Logical predicates (*and*, *or*, and *not*) are defined in a conventional manner to concatenate multiple predicates.

dense(Group G, threshold) <=>
Relative distance between any two group members < threshold
sparse(Group G, threshold) <=>
Relative distance between any two group members > threshold

Therefore, CFG syntax to represent necessary relationships of a group activity is defined using predicates. Note that the special time interval 'this' is used to specify the temporal relationship between the defining group activity itself and its other sub-events.

Relationship
-> Logical-Predicate "(" Relationship "," Relationship ")"
| Temporal-Predicate "(" "'this'" "," time_var ")"
| Temporal-Predicate "(" time_ var "," "'this'" ")"
| Temporal-Predicate "(" time_var "," time_ var ")"
| Individual-Spatial-Predicate "(" person_var "," person_var "," int ")"
| Group-Spatial-Predicate "(" group_var "," int ")"
Individual-Spatial-Predicate -> "near" | "touch"
Group-Spatial-Predicate -> "dense" | "sparse"

As a result, the full representation is composed of three main parts: a list of member variables *MemberVariable-Defs*, a list of time intervals of sub-events *TimeIntervalDefs*, and a list of relationships *Relationship*. Participants, member variables, and time intervals defined through *participants, MemberVariableDefs,* and *TimeIntervalDefs* are used in the term *Relationship* to describe necessary relationships. Three terms are integrated in our final CFG syntax where *GroupActivityDefine* is the starting variable. Example representations of the group activity 'a group of people are carrying a large object by command of another person' and 'group fighting' are presented as well.

GroupActivityDefine
-> name "(" participants ")" "=" "{" MemberVariableDefs "," TimeIntervalDefs "," Relationship "};"
Ex> ***CarryByCommand**(Group G1, Person p1) = {*
∀ a in G1,
*list(def (t1, **Carry**(a)), def(t2, **Command**(p1))),*
and(equals(t1, this), meets(t2, t1))
};
***GroupGroupFighting**(Group G1, Group G2) = {*
∀ a in G1, ∃ b in G2,
*list(def(t1,**Approach**(G1, G2)), def(t2,**Fight**(a, b)),*
and(and(dense(G1), dense(g2)),
and(equals(t1, this), meets(t1, t2)))
};

A group activity can always be decomposed into four elementary types if and only if member variables can be divided into independent pairs and/or singles. That is, we limit a member variable to have interaction with only one other variable to make the recognition process tractable.

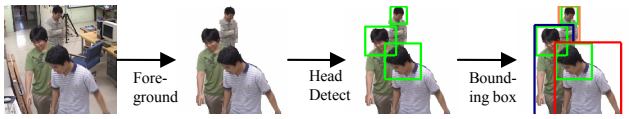

Figure 2: Low-level processing of the system.

3. Recognition

This section discusses an algorithm to recognize high-level group activities that have been represented using the CFG syntax. Our recognition process conveys the hierarchical structure of our group activity representation, recognizing group activities based on the recognition results of their sub-events. Once candidate time intervals of sub-events are detected, the system searches for valid combinations of time intervals that satisfy temporal constraints of the group activity. Next, the system checks whether persons who performed the sub-events can form correct groups or not. Only when valid groups satisfying the spatial constraints can be constructed with participants of the sub-events, the system is able to deduce that the group activity occurred with the sub-events.

3.1. Base case: recognition of individual activities

The base case of the group activity recognition is the recognition of individual activities. High-level group activities are represented in terms of activities of individual persons and other simpler group activities (which themselves can be decomposed as well), suggesting that the recognition of human actions and human-human interactions must be performed first. We in our previous work have presented an activity recognition methodology which is able to recognize human-humans interactions such as a 'fighting' [8]. In this subsection, we construct new low-level components of the system for object recognition and motion estimation, in order to enable the system to detect human interactions (i.e. base cases) more reliably with our previous human activity recognition system.

For each frame, the low-level of our system first performs background subtraction to segment foreground regions. Since one foreground region may contain multiple persons due to their occlusions, our system takes advantage of head detection using object detector developed by Viola and Jones [9]. Persons' bounding boxes are placed considering foreground regions as well as positions of detected heads. This person-segmentation method is similar to that of W4 [4], which also takes advantage of head detection for segmenting occluded persons. Once a person is correctly segmented, color histograms are used to classify the type of the person (e.g. policeman vs pedestrians), if needed. Viola and Jones's detector [9] is also used for objects (e.g. laptop computer). Hidden Markov models are constructed to estimate motion of each individual, where width/height ratio and the center position of a bounding box are used as features for the HMM. These results are passed to the higher-level of the system, recognizing human actions and interactions.

```
CONSTRAINT_CHECK(Activity a) {
   Let tr be a tree where time interval variables are
      node and temporal predicates are edges;
   Node r = root node of tr;
   ASSIGN(r);
   if (all nodes of tr is assigned) return true;
   else return false;
}
ASSIGN(Node n) {
   Node p = n's parent;
   List a = List of candidate time intervals that can be
            assigned to n;
   if (p==null) n.a = a;
   else{ for (each time interval t in a)
            for (each time interval pt in p.a)
               if (t and pt satisfies temporal relationship)
                  add t to n.a; }
   for (each node c who is a child of n) ASSIGN(c);
}
```

Figure 3: Pseudo code of the temporal constraint check algorithm

3.2. Hierarchical temporal constraint matching

The recognition of group activities is done using a hierarchical algorithm from bottom to top. We discussed the recognition of base cases in the previous subsection. For all non-base cases, i.e. all group activities constructed based on individual activities, the sub-events of a group activity are recognized first and temporal relationship among detected time intervals are analyzed next. Once a group activity is recognized, the results can hierarchically be used for the recognition of a higher-level group activity that contains the activity as its sub-event.

The problem of checking whether detected time intervals satisfy a temporal relationship or not is one of the traditional constraint satisfaction problems. An activity can occur multiple times, suggesting that each sub-event has multiple valid time intervals. Therefore, if r is the average number of time intervals of one sub-event and n is the number of sub-events, there are r^n possible combinations of time interval associations. The goal of the system is to search for combinations that satisfy the temporal relationship of a represented group activity.

In order to detect such combinations efficiently without spending an exponential amount of computations, we model the relationship of a group activity as a set of trees: We first enumerate relationships to make them in DNF (disjunctive normal form). Each clause of DNF is a conjunction of temporal relations, and we construct an undirected graph for each clause where variables indicating time intervals are nodes and predicates between them are edges. In the case when temporal relationships for a group activity contain a cycle, our system removes the cycle (i.e. converts to a tree) to perform the recognition process,

which is an approximation of the actual temporal constraints. With the assumption of a tree structure of temporal relationships, searching for a valid combination can be done in polynomial time. Figure 3 shows detailed pseudo code of our time interval detection algorithm. For each tree, our algorithm searches for valid combinations by assigning time intervals to nodes (i.e. time variables) from the root to leaves.

The algorithm treats sub-events done by any persons as valid candidate time intervals as long as they satisfy the temporal constraints. However, in order for a group activity to occur, the sub-events associated with the same member variable (or same participant of the group activity) must be done by the same person. Therefore, our system discards time interval combinations which violate the constraint that 'sub-events associated with the same member variable must be done by the same person'.

As a result, valid combinations are detected, and individuals are assigned to a member variable for each combination. Individuals who performed a sub-event are assigned to the member variable corresponding to the actor of the sub-event. We must note that more than one person can be assigned to each member variable, since multiple persons can perform an identical sub-event in the same time interval. In the case when a sub-event is an interaction between two member variables, a person assigned to one member variable may depend on a person assigned to the other member variable. In this case, instead of assigning persons to each member variable independently, our system assigns pairs of persons to two member variables jointly.

Once valid time intervals of sub-events are calculated, time intervals of a group activity itself can also be computed by calculating the range of the special time interval 'this'. The time complexity of the overall algorithm is $O(r^2 + m)$ where m is the total number of combinations satisfying temporal constraints.

3.3. Group actions and group-persons interactions

Detected time intervals of sub-events form a group activity only when their temporal constraints are satisfied, and only when a group can be formed with persons assigned to member variables. If a valid group can be formed, our system can conclude that the group activity occurred in the calculated time interval with a group of individuals who executed the sub-events. That is, our system is defining a group as a set of individuals who satisfy the representation of a group activity, and recognizing the group activity by searching for such set. Therefore, in this subsection, we present an algorithm to calculate a valid group from individuals assigned to member variables of a group action or a group-persons interaction.

For each member variable, the system computes the

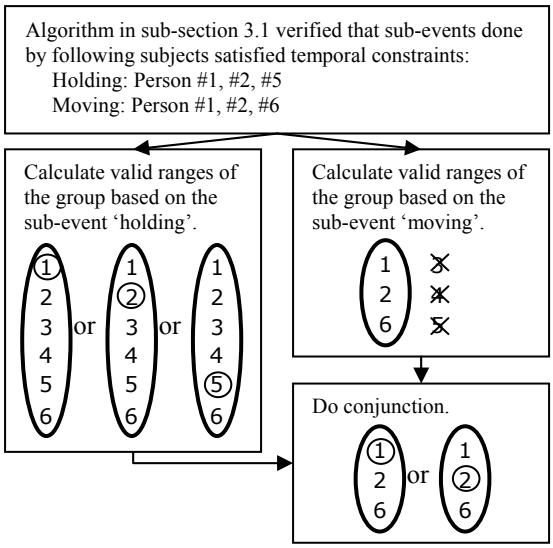

Figure 4: Example process flow of the recognition of group action 'group carry'. The 'group carry' is represented as all group members ($\forall$) 'moving' into one direction while any of group member ($\exists$) 'holding' the object.

range of a valid group so that it includes persons assigned to the member variable or excludes persons not assigned, depending on the type of the quantifier. If the quantifier attached to the member variable is an existential quantifier ($\exists$), the range of a group is decided to be any set of individuals that contains at least one of the persons who are assigned to the member variable in a given time interval. If the quantifier is a universal quantifier ($\forall$), the group must be a subset of all individuals assigned to the member variable, implying that the maximum range of a valid group is the set of all individuals assigned.

We represent the range of a group calculated for each member variable as follows, in terms of a set with tags attached to its member. For a member variable with an existential quantifier, the valid group is represented as a set of entire persons with a uniform tag attached to the persons who are assigned to the member variable. That is, if there are six persons, and person #1, person #2, and person #5 are assigned to one member variable, the valid range of it is represented as $\{1^c, 2^c, 3, 4, 5^c, 6\}$. Any subset of the represented set can become a valid group, if and only if it contains at least one member with the tag. If a universal quantifier is attached to the member variable, the range is simply represented as a set of persons assigned to the variable. For example, if person #1, person #2, and person #6 have been assigned, the valid range is represented as *{1, 2, 6}*. Again, we can see that any subset of a represented set is a valid group.

Once a valid range of the group has been represented for each member variable, the system combines all of them to calculate the final range. The system computes a conjunction of ranges calculated by the member variables,

since all member variables belong to one group in case of a group action or a group-persons interaction. That is, the system is detecting the group that is within the ranges calculated for member variables. For example, if the range of one member variable of a group action is represented as $\{1^c, 2^c, 3, 4, 5^c, 6\}$ and that of the other member is $\{1, 2, 6\}$, the conjunction of them $\{1^c, 2^c, 6\}$ represents the total range of a valid group. The example process of our algorithm is presented in figure 4. If two ranges calculated with existential quantifiers such as $\{1^c, 2^c, 6\}$ and $\{1, 2^d, 6^d\}$ are combined, two tags need to be maintained as $\{1^c, 2^{c,d}, 6^d\}$. Any subsets containing both tags *c* and *d* are valid.

If a group action or a group-persons interaction contains simpler group activities as its sub-event, the system again computes the conjunction of ranges calculated based on the member variables and a range of the group performing the sub-event. The overall computation can be done in $O(k)$ where k is the number of persons appearing in the scene.

Spatial constraints are further applied to get the final group who has performed the represented group activity. For example, if the group must satisfy the spatial constraint 'dense', the represented range will be divided into several clusters that are spatially separated.

3.4. Group-Group and inter-group interactions

Unlike group actions or group-persons interactions, sub-events of group-group interactions can be associated with two member variables of two different groups. This suggests that the valid range of one group is dependent on the range of the other group. In order to calculate the valid ranges of two groups based on member variables of a group-group interaction, our system decomposes a list of member variables into multiple independent pairs and singles. Because of the characteristics of our representation, one member variable is dependent at most on one other variable, which makes the division of the full list of member variables into several independent pairs possible. Our system analyzes valid ranges of two groups for each pair, and then combines them by finding a conjunction of ranges calculated.

We present an algorithm to calculate and represent valid range of each pair of two dependent groups. Types of quantifiers attached to two member variables decides characteristics of valid ranges, dividing them into four different cases: $\exists\,\exists$, $\forall\,\exists$, $\forall\,\forall$, and $\exists\,\forall$. A method to calculate a conjunction of calculated ranges is also presented. After calculating the final ranges of two groups, spatial relationships are checked as in subsection 3.2. Inter-group interactions can be interpreted as a special case of group-group interactions, where two participating groups must be an identical group.

Case 1: $\exists\,\exists$. The algorithm presented in subsection 3.1 assigns pairs of persons to pairs of two dependent member variables. In the case when two existential quantifiers are attached to two member variables, at least one assigned pair of persons has to be allocated for two groups by definition. For example, suppose the 'group trading' activity of two groups are defined as having a sub-event of 'trade' interaction between at least a pair of individual members from each group: If persons (#1, #5) and (#4, #6) are pairs of persons who performed the sub-event 'trade', then (G1 must contain person #1 and G2 must contain person #5) or (G1 must contain #4 and G2 must contain #6). Our system represents these ranges of two groups in terms of an array, where tags similar to that of section 3.2 are attached to cells of an array. In the above example, only S[1,5] and S[4,6] of the array S will be set to the tag value *c*.

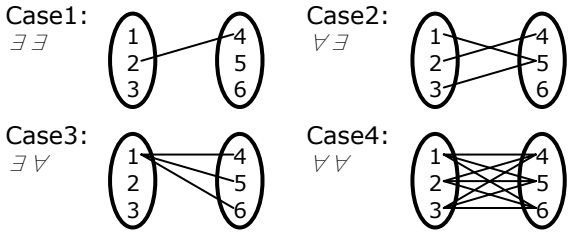

Figure 5: Diagram illustrating member relationships in 4 cases.

Case 2: $\forall\,\exists$. This is the case where all members of the first group must have more than one corresponding person in the other group. That is, any person *i* of the first group has some corresponding person *j* in the other group, where the pair (i, j) is an assigned pair of two member variables. For example, a group activity representing all members of one group attacking someone in the other group is this case. Our system represents a valid range of two groups as two sets with specific tags attached to members, similar to the case of 'group action'. While the first group is represented as a set of members who has at least one corresponding persons in the other group, the other group is represented as a set of member with tags indicating the corresponding person in the first group. For example, if (1, 3), (1, 5), and (2, 4) are assigned pairs of variables, the valid range is represented as $\{1, 2\}$ and $\{3^{c\_1}, 4^{c\_2}, 5^{c\_1}, 6\}$. Any pair of subsets of two sets is a valid group, if and only if the following constraint is satisfied: For each member in the first group, at least one member in the second group must have the tag corresponding to it. For example, $\{1\}$ and $\{3, 6\}$ is a valid pair of groups, while $\{1, 2\}$ and $\{3, 6\}$ is not since it is missing c_2.

Case 3: $\forall\,\forall$. In this case, a pair of any member of the first group and any member of the second group must be an assigned pair of two member variables (e.g. group-group fighting). Our system uses a greedy heuristic approximation to represent all possible ranges of valid groups. The system finds multiple candidate pairs of sets that do not overlap each other. This is done by iteratively detecting a valid pair of sets which makes any pair of their

subsets to be valid as well. For example, if two groups {1, 2} and {3, 4, 5} are detected to be one of the valid ranges, their subsets {1} and {3, 5} also forms a valid pair.

Case 4: $\exists\,\forall$. Case 4 can be thought as a soft version of case 3. In case 4, only one member in the first group needs to be associated with all members in the second group. An identical algorithm can be applied for the detection of valid ranges. The first group is represented as a set of all members, and a tag attached to the members who are associated with all members in the second group. The second group is simply represented as a set of members associated with the member in the first group. For example, if person #1 and person #2 are associated with all members in {3, 4, 5}, it can be represented as $\{1^c, 2^c, 6\}$ and {3, 4, 5}

Combinations. We take a conjunction of decomposed pairs of member variables. As a result of a conjunction, the system maintains two types of representations. An array representation generated by case 1 and set representations generated by cases 2, 3, and 4. Only one array is needed to represent conjunctions of multiple arrays produced by case 1. Case 2 always generates only one pair of ranges. On the other hand, case 3 and case 4 generate multiple candidate pairs of two sets as a valid range, suggesting that their conjunction also generates multiple pairs. The maximum number of set representations constructed as a result of conjunction is k^2 where k is the number of persons in the scene, since sets constructed in case 3 and 4 do not overlap each other. Therefore, the complexity of calculating valid range of groups is $O(k^2)$. When calculating a conjunction of ranges, tags attached to members of sets must always be preserved. For example, if pairs of ranges $\{1, 2\}$-$\{3^{c\_1}, 4^{c\_2}, 5^{c\_1}, 6\}$, $\{1, 2\}$-$\{3^{d\_1}, 4^{d\_1}, 6^{d\_2}\}$, and $\{1^e, 2, 7\}$-{3, 4, 5, 6} are combined, the result is $\{1^e, 2\}$-$\{3^{c\_1, d\_1}, 4^{c\_2, d\_1}, 6^{d\_2}\}$. That is, {1}-{3, 4} is a valid pair of groups, while {1, 2}-{3, 4} is not since the second set is missing *d_2*.

4. Experiments

We implement the system presented in this paper, and test it to recognize high-level group activities such as 'group stealing' and 'group arresting'. Notably, we are using CCTV videos that have been downloaded from *YouTube* as well as videos that we have taken in various environments. We implement and test our group activity recognition system for various types of group activities, while measuring the performance of our system compared to the previous recognition approach.

We have represented seven different types of group activities, and tested the system with videos downloaded from *YouTube* and videos taken with total of six participants in various environments. 'Group move', 'group carry', 'group carry by signal', 'group fighting', 'inter-group fighting', 'group stealing', and 'group arresting' are the activities tested. 'Group move' indicates a group of people moving in the same direction and 'group carry' describes a group of people carrying a table or other large objects. We already have defined and represented 'group carry by signal' and 'group fighting' in section 2.2. 'Inter-group fighting' is an inter-group version of group fighting. 'Group stealing' is a complex group-group interaction where one of thieves is stealing an object (e.g. laptop) while other thieves are distracting a group of owners of the object. 'Group arresting' indicates the situation where policemen are arresting a group of criminals. A total of 35 sequences, five videos per group activity, are tested to measure the performance.

The videos were taken in 15 frames per second in the resolution of 320 *240. As a result, approximately 8000 frames were obtained from 35 sequences. We have randomly chosen 800 frames of sub-sequences, in order to train the object detector (i.e. head detector) and HMM for motion estimation. The representation of group activities is encoded by a human expert, following our representation scheme. For example, the representation of 'group stealing', a complex group activity with 3 quantifiers, is as follows:

```
GroupStealing(Group Thieves, Group Owners) = {
    ∃ a in Thieves, ∀ b in Owners, ∃ c in Thieves,
  list( def(t1, Approach( Thieves, Owners)),
        list( def(t2, TakeObject(a)),
              def(t3, Distract(c, b))) ),
  and( and( before(t1, t2), during(t2, t3)), equals(t2, this) )
};
```

Once the low-level part of our system is trained and the representation is encoded by the human expert, our group activity recognition system behaves fully automatically. The system was tested on all 35 sequences.

Figure 6 shows the example sequences of group activities which our system successfully recognized. Bounding boxes have been drawn for each person. Groups detected as a result of our algorithm are indicated using the color of bounding boxes. Figure 7 shows example time interval recognition results of the top-most sequence, the *YouTube* downloaded video of 'group stealing'.

Table 1 illustrates the final recognition accuracy of our algorithm. The type of each group activity is specified: GA stands for 'group action', GP for 'group-persons interaction', GG for 'group-group interaction', and 'IG' for 'inter-group interaction'. Types of quantifiers attached to member variables of each activity are also listed. The performance is compared with a system implemented following the ***previous method***. The ***previous method*** is our implementation of group activity recognition system following the previous paradigm. Previous systems designed for the recognition of simple group activities generally recognizes groups using spatial information first and then analyzes their activities next [6,11]. In order to show the advantage of our recognition algorithm over previous systems, we have implemented another version of

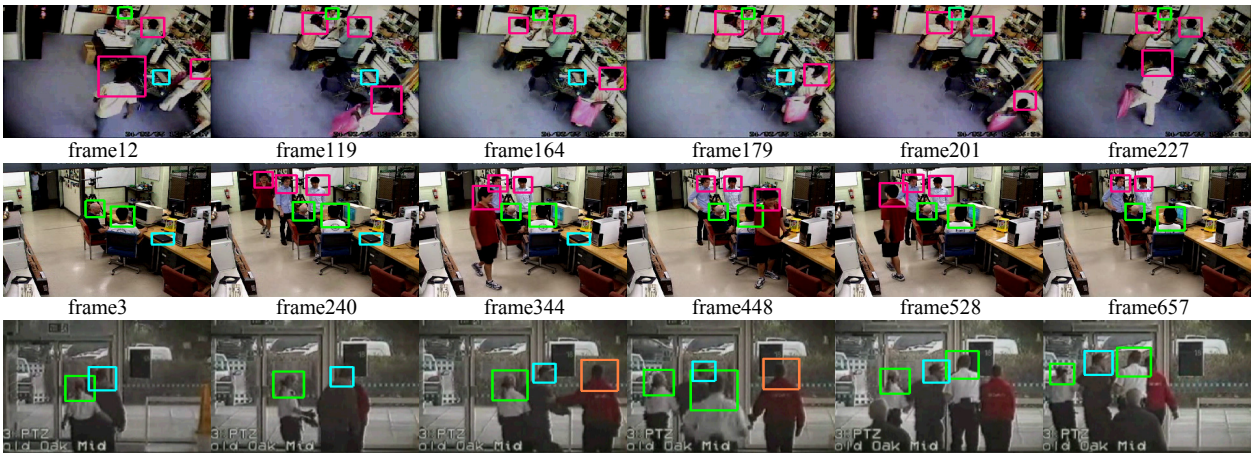

Figure 7: Processed video sequences of group activities. The top-most sequence and the middle sequence are example videos of 'group stealing'. The bottom-most sequence is an example video of 'group arresting'. The top sequence and the bottom sequence are from real CCTV videos that have been downloaded from *YouTube*, and the middle sequence is from the videos that we have taken in an office environment. In case of 'group stealing', red bounding boxes are used to denote thieves, while green bounding box are used to denote owners. A cyan bounding box is used to label the object, a laptop. In case of 'arresting', green boxes indicate policemen, and cyan boxes indicate persons being arrested. (This figure is best viewed in color.)

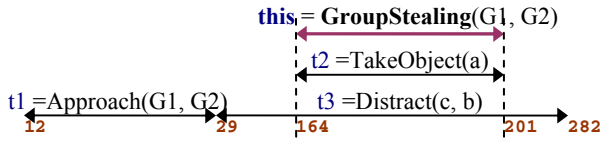

Figure 6: Example time interval detection results of 'stealing'.

our system called '***previous method***' which uses identical representation but follows previous recognition paradigm.

Table 1 shows true positive rates. False positive rates were almost 0 in all cases with both systems, since recognizing multiple sub-events satisfying the specific relationship by 'mistake' is extremely unlikely. The recognition accuracy depends on the inherent characteristics of the group activity. We are able to observe that our system performs superiorly over the ***previous method***. The ***previous method*** did not perform well especially when recognizing 'group stealing', since spatial distance among thieves was changing over time.

Activity\System	Type	Qntfs.	Prev. Method	Our System
Move	GA	$\forall$	5/5	5/5
Carry	GA	$\forall$	3/5	5/5
Carry by signal	GP	$\forall$	3/5	4/5
Fight	GG	$\forall\exists$	3/5	4/5
Fight	IG	$\exists\exists$	4/5	4/5
Steal	GG	$\exists\forall\exists$	2/5	4/5
Arrest	GG	$\forall\exists$	4/5	4/5
total			**24/35**	**30/35**

Table 1: Recognition accuracy of the system

5. Conclusions

We presented a novel representation and recognition algorithm for the recognition of high-level group activities. The technical contributions of this paper are the representation scheme to represent various types of group activities, and the new hierarchical algorithm for the recognition. We presented recognition methodology for group activities with complex temporal, spatial, and logical structures, which has not been developed previously.

References

[1] J. F. Allen and G. Ferguson. Actions and Events in Interval Temporal Logic. J. of Logic and Computation, 4(5), 1994.

[2] F. Cupillard, F. Brémond and M. Thonnat, Group Behavior Recognition With Multiple Cameras, WACV 2002

[3] S. Gong, and T. Xiang. Recognition of group activities using dynamic probabilistic networks. ICCV (2): 742-749, 2003.

[4] I. Haritaoglu, D. Harwood, and L. Davis. W4: Real-Time Surveillance of People and Their Activities. IEEE T PAMI, 22(8):809-830, August 2000.

[5] S. Hongeng, R. Nevatia, and F. Bremond. Video-based event recognition: activity representation and probabilistic recognition methods. CVIU, 96(2):129–162, 2004.

[6] S. M. Khan, and M. Shah. Detecting group activities using rigidity of formation. ACM Multimedia: 403-406, 2005.

[7] F. Lv, J. Kang, R. Nevatia, I. Cohen, and G. Medioni. Automatic tracking and labeling of human activities in a video sequence. PETS 2004.

[8] M. S. Ryoo and J. K. Aggarwal, Recognition of Composite Human Activities through Context-Free Grammar based Representation, CVPR 2006.

[9] P. Viola and M. J. Jones, Rapid Object Detection using a Boosted Cascade of Simple Features, CVPR 2001.

[10] V.-T. Vu, F. Bremond, and M. Thonnat. Automatic video interpretation: A novel algorithm for temporal scenario recognition, IJCAI-03

[11] D. Zhang, D. Gatica-Perez, S. Bengio, and I. McCowan. Modeling individual and group actions in meetings with layered HMMs. IEEE T MM, 8(3):509-520, June 2006.