

# Detecting Interleaved Sequences and Groups in Camera Streams for Human Behavior Sensing

Athanasios Bamis, Jia Fang and Andreas Savvides  
Embedded Networks & Applications Lab (ENALAB)  
Yale University, New Haven, CT 06520, USA  
{firstname.lastname@yale.edu}

**Abstract**—Deployments of camera security systems are capable of capturing long data sequences about human activity. This paper deals with processing of detected sequences at a more macroscopic level to detect *chains of events* based on a prior given specification. In our problem, sensed interactions between people are modeled as sequences of pairwise events that are interleaved with other interactions taking place in the background. We formulate the problem as an isomorphic subgraph matching problem and solve it to detect a chain of events, its participants and their roles. We further evaluate our solution in the presence of background interference from other interactions and give analytical and empirical results about the performance of our algorithm.

## I. INTRODUCTION

Many types of sensor systems today can deliver spatio-temporal sequences that can be used in the study of group formation and behaviors of people for security, elder monitoring, social interactions etc. Camera networks and video analytics in particular, can provide human locations associated with a timestamp and potentially can recognize humans across multiple non-overlapping views [1], [2]. As sensing continues to mature, the application of such networks in security settings becomes more attractive and creates an immediate need for schemes that detect patterns at a more macroscopic level over larger spatial and temporal intervals. Such schemes could have immediate use in security applications, elder monitoring and could potentially be applied to other types of sensors such as GPS sensors which are part of the up and coming participatory sensing applications. The advantage that networks of cameras and other sensors hold over their human counterparts is the ability to catch chains of events that span over larger temporal and spatial scales. Conspiratory behavior is a good example that falls under this umbrella.

A scenario of such a complex activity evolving over a large time span and in a large space can be inspired by the recent (2008) blockbuster movie “21”. This movie presents (with some artistic license) the story of the “MIT Blackjack Team”. A conspiring group of players that used card counting techniques in order to collectively beat the Blackjack game. In the movie each player has one of two distinct roles. “Counters”, which are responsible for identifying “interesting tables” and “big players” who are signaled by the counters to make large bets only in interesting tables. These roles are used in order to prevent the casino personnel, depicted in the movie as a group of people looking at the streams from a large number of surveillance cameras, from identifying their group

and the playing pattern created by counting cards, something which could ban them from the casinos. Members of the group, thus, avoided direct contact and limited themselves to only short interactions with a limited number of other members of the group. The casino example, although far fetched in terms of sensing, reveals some of the key requirements of our approach. A group of conspiring entities execute a sequence of moves to carry out a task while hiding their association to remain undetected.

In this paper we assume that camera sensing reliably detects interactions between individuals, and we focus on investigating how to detect groups from sensor data with the goal of detecting chains of pair-wise events that are interleaved with other background interactions in the stream of sensor data. We anticipate that this approach can consume the state-of-the-art results in computer vision that can detect interactions at several level of granularity, but it is also directly usable with other types of readily available sensors. To use the solution, the expert observer can provide a full or partial specification of the chain of events using knowledge of the sequence of actions and roles needed to carry out the act. Our objective is to use this specification to identify the sequence in the data stream and thus reveal the identities and roles of the people involved. Our solution formulates and solves the problem using our data representation to pose the problem as an isomorphic subgraph matching problem and characterizes the performance of the solution in the presence of interference from background interactions. In its initial form this approach is targeted for applications that can sense the location of humans over time such as security, elder monitoring or participatory sensing.

The rest of this paper is organized as follows: Section 2 discusses our sensing model and how to detect pair-wise events from data streams. Section 3 states the problem of detecting interleaved subsequences based on a given specification. In section 4 we formulate and solve this problem as an isomorphic subgraph matching problem. Section 5 derives an analytical result for the effect of background interference. Section 6 discusses our simulation and modeling efforts and benchmarks our solution. Finally section 7 surveys the related work and section 8 concludes the paper.

## II. SENSING MODEL

In the rest of the paper we will present our solution using a simplified sensing model for detecting pairwise events. This



Fig. 1. A drug deal as a simple scenario of a behavior description as a sequence of interactions.

sensing model can be applied to systems that provide human locations on a map/floorplan, also including camera sensor networks. To experiment with this concepts we have also developed a separate solution that uniquely identifies people in the camera’s field of view by correlating information between camera detections and wearable accelerometers [anonymous]. In this paper we omit the details of the sensing approach and focus on the detection of chains of events. To give a tangible example, in this paper we consider a drug-deal scenario, simulated using Bohemia Interactive’s Virtual Battle Space 2 (VBS2) simulator, used mainly for military simulations. A video rendering of our simulation can be viewed online at <http://www.youtube.com/watch?v=eFPEXgIXAaU>.

The drug deal scenario takes place in a busy scene. This scenario includes four actors or protagonists, namely the “customer”, the “cashier”, the “runner” and the “holder”. The customer will first visit the cashier, make a deal and exchange money with him. Following, the cashier will signal the runner, who will pick up the drugs from the runner and deliver them back to the customer (Figure 1).

One possible representation of a pair-wise event between two persons can be created by considering the spatial and temporal proximity. This can be defined as a threshold between locations for a given time period. In the same manner, we can define pair-wise events involving a person and a location (person inside or outside the location) or object (proximity of the person to the object).

To extract these events from a time-stamped location data set we need to define a proper data structure that can capture the various associations taking place in the monitored scene. A natural representation will be an *association graph*, where vertices are the various entities in the monitored scene and edges indicate interactions among these entities.

An entity can be anything of interest for the particular monitored scene and application. An entity can, thus, be a person, an object, a location or an observed event (e.g., exchange of objects, phone call). In the case of people the association can be used to indicate some form of interaction or some prior knowledge that two people are related. An edge between a person and an object can be used to represent an interaction of the person with the object and an edge between a person and a location can be used to represent the presence of a person at a particular location. In the case of observed events we can have an edge indicating the participation of a person in the event, something which can potentially be prior knowledge. However, note that locations and objects have to be selected carefully in order to avoid combinatorial explosions in the number of interactions between entities (e.g., including the entrance of a building as an entity will create an indirect

association between any two persons in the building).

Since an association graph is something static, it can refer only to a single time instant of the evolution of the monitored scene or in a small time interval. In this small time interval we assume that the entities and their associations do not change. The smallest time interval would be a single frame. However, if we attempt to create an association graph for every frame in the video we will inevitably introduce a large amount of noise, since we will consider as interaction between two persons, for instance, any physical proximity regardless of its duration (imagine two persons passing by each other). Thus, we need to define a *sampling window*, which will be the time interval  $t_{smp}$  for which an association graph is defined. For the duration of the sampling window we assume that the scene remains static and we consider all the entities and interactions that occurred in the scene. Thus, we have the following definition for a dynamic association graph:

*Definition 1 (Dynamic Association Graph):* A time series of association graphs defined at consecutive sampling windows. More formally, a time-varying graph  $G(t_i)$  that is defined for  $t_i \in \{t_0, t_1, \dots, t_n\}$ . If sampling starts at time  $T_{start}$  and our sampling window is  $t_{smp}$ , then  $t_i = T_{start} + i \cdot t_{smp}$ . The set of vertices  $V(t_i)$  (edges  $E(t_i)$ ) includes all the vertices (edges) of the association graph  $G_{i-1,i}$  defined in the interval  $(t_{i-1}, t_i]$ , namely  $V(t_i) = V(G_{i-1,i})$  ( $E(t_i) = E(G_{i-1,i})$ ).

Apparently, in a dynamic monitored scene we can only maintain a dynamic association graph for only a limited amount of time, using a sliding time window  $T_{dwin} = \{t_0, t_1, \dots, t_n\}$ .

### III. PROBLEM DEFINITION

Our goal is to recognize a chain of events involving multiple participating entities by rearranging the data to extract a stream of pairwise events as described in the previous section. In this event stream, the associations of interest will appear interleaved with other associations occurring in the scene, that manifest themselves as background noise in our problem. The main challenge in our problem is to provide a solution for detecting sequences that match a given specification, identify the people involved and characterize the performance of the solution with respect to background noise contributed by other interacting entities.

In the following, let  $\mathcal{E}$  denote the set of entity IDs. The input is a sequence  $\mathcal{I}$  of pair-wise events ordered according to the start time of event occurrence, i.e.  $\mathcal{I} = PE_1, PE_2, \dots, PE_N$ , where  $PE_i \in \mathcal{E} \times \mathcal{E}$  for all  $i \in \{1, 2, \dots, N\}$ . Given two pair-wise events  $x$  and  $y$ , we say that  $x$  and  $y$  satisfy a *chaining constraint*, and we write  $x \sim y$ , if there is exactly one entity ID which is common to both  $x$  and  $y$ . As an example, suppose we are given entity IDs  $\{E1, E2, \dots, Ek\}$ , and pair-wise events  $x, y, z$  where  $x = (E1, E5)$ ,  $y = (E2, E5)$  and  $z = (E2, E7)$ . By definition, we have that  $x \sim y$  and  $y \sim z$ , but  $x$  and  $z$  can’t be chained. We also extend the notion of chaining to more than one pair-wise events, in which case all the pair-wise events in the chain are defined to have *exactly one entity in common*. For example, if  $w = (E5, E3)$  then  $x \sim y \sim w$ . Note that

whether two pair-wise events satisfy a chaining constraint is independent of the ordering of the entity IDs in each event. This is important because in the problem that we consider pair-wise events  $(E_i, E_j)$  and  $(E_j, E_i)$  are considered equivalent.

We define an *event variable*  $x$  to be a variable which can take on any value in the set  $\mathcal{E} \times \mathcal{E}$ . We assume the interaction patterns among entities participating in a behavior is known, while the specific IDs of the participating entities are *not* known. Roughly speaking, we are interested in behaviors involving multiple entities which can be described as a sequence of pair-wise events occurring over time. Interaction patterns among the participating entities are captured through pair-wise events which have entities in common, or pair-wise events which have no entities in common. Such behaviors admit a natural and intuitive encoding as a sequence of event variables together with chaining constraints.

The key item of interest here is that behaviors are specified using event variables as opposed to simply pair-wise events since the IDs of the participating entities are unknown. Formally, a behavior consisting of  $n$  pair-wise events is specified as a sequence of  $n$  event variables  $\mathcal{S} = x_1, x_2, \dots, x_n$  together with a set of chaining constraints denoted  $\mathcal{C}$ . We call  $\mathcal{S}$  and  $\mathcal{C}$  the *specification* of the behavior, and we call  $\mathcal{S}$  the *search sequence* of the specification. In the following, we will say that a subsequence of the input sequence  $\mathcal{I}$  is any sequence obtained by removing zero or more elements of  $\mathcal{I}$ . Given any subsequence of  $n$  pair-wise events  $e_{i1}, e_{i2}, \dots, e_{in}$  of the input sequence, we say that the subsequence *satisfies* or *matches* the specification of the behavior if all chaining constraints are satisfied when  $x_k = e_{ik}$  for all  $k \in \{1, \dots, n\}$ . Note that the temporal properties of the behavior is captured in its specification by means of the ordering of the sequence  $x_1, x_2, \dots, x_n$  and the ordering of the subsequences of  $\mathcal{I}$ . Hence, that  $e_{i1}, e_{i2}, \dots, e_{in}$  is a satisfying (or matching) subsequence does not necessarily imply  $e_{i2}, e_{i1}, \dots, e_{in}$  is a satisfying subsequence, since the chaining constraints may not be all satisfied when  $x_1 = e_{i2}$  and  $x_2 = e_{i1}$ .

We now give a formal statement of the problem. Given the input sequence  $\mathcal{I}$  and the specification of a behavior, the aim is to enumerate all subsequences  $\mathcal{I}'$  of  $\mathcal{I}$  where  $\mathcal{I}'$  satisfies the specification of the behavior. If a subsequence  $\mathcal{I}'$  of  $\mathcal{I}$  satisfies the specification of the behavior, then clearly entities occurring in the pair-wise events of  $\mathcal{I}'$  are behaving in a manner consistent with the behavior of interest (as encoded by its specification) in the time period over which the events of  $\mathcal{I}'$  occur. Hence, by determining all satisfying subsequences, we not only identify entities which are participating in a behavior of interest, but also when those behaviors occur. *The key challenge of this problem is that the IDs of the entities carrying out a behavior of interest are not known beforehand, and must be determined from a sequence of spatiotemporal data that may consist of thousands of pairwise interactions among hundreds of entities.*

#### IV. DETECTING SIMPLE GROUP BEHAVIORS

In this section, we show that the behavior detection problem defined in the previous section can be reformulated via a graph based model as a subgraph isomorphism problem. Roughly speaking, the specification of the behavior of interest is first converted to an undirected graph with labeled edges in which the event variables correspond to the edges, and the order of the events are enforced through the edge labels. The input sequence is also converted to an undirected graph with labeled edges in which the vertices correspond to entities, and each edge corresponds to a pair-wise event in the input sequence labeled with the index of the event in the input sequence. Once the behavior specification and input sequence are both converted to graphs, the behavior detection problem becomes the well-known *subgraph matching* problem where the idea is to enumerate all subgraphs of a graph which are isomorphic to a given graph.

##### A. Converting To Subgraph Matching

The intuition for converting our original problem to subgraph matching is based on the notion of isomorphism. Two graphs  $G_1$  and  $G_2$  are isomorphic if  $G_1$  and  $G_2$  have the same number of vertices and edges and there exists a one-to-one correspondence  $f$  from the vertices of  $G_1$  to the vertices of  $G_2$  such that any two vertices  $u$  and  $v$  of  $G_1$  are adjacent in  $G_1$  if and only if  $f(u)$  and  $f(v)$  are adjacent in  $G_2$ . Intuitively, two graphs are isomorphic when they have the same “structure” as expressed through edges regardless of the vertex labels. Hence, isomorphism addresses two of our main difficulties in searching for a match to the specification of a behavior of interest, namely the notion of chaining between pair-wise events, and the abstract relations between entities.

Suppose the specification of a behavior is given with search sequence  $\mathcal{S}$ . By expressing each event variable in the specification of a behavior as an edge of a graph, we can construct a graph  $G_{\mathcal{S}}$  that captures all the chaining constraints of the specification without requiring that the IDs of the participating entities be known. By expressing in a similar manner our input sequence  $\mathcal{I}$  as a graph  $G_{\mathcal{I}}$  whose edges are the pair-wise events in  $\mathcal{I}$ , subsequences of  $\mathcal{I}$  that satisfy the specification will appear as subgraphs of  $G_{\mathcal{I}}$  that are isomorphic to  $G_{\mathcal{S}}$ .

We define an *entity variable* to be a variable which can take on any value in the set of entity IDs  $\mathcal{E}$ . By definition, each event variable consists of a pair of entity variables. An entity variable of  $\mathcal{S}$  is any entity variable which appears in some event variable of  $\mathcal{S}$ . We say that an entity variable of  $\mathcal{S}$  is *free* if there are no chaining constraints on the event variable it belongs to, and we say that an entity variable is *dependent* if it belongs to an event variable which is part of at least one chaining constraint. The graph  $G_{\mathcal{S}}$  is defined as follows.

- 1) Add a vertex for every chaining constraint, representing the dependent entity variable.
- 2) Add a vertex for every event variable appearing in only one chain, representing the free entity variable of the event variable, and an edge between this vertex and the dependent entity variable of the chain.

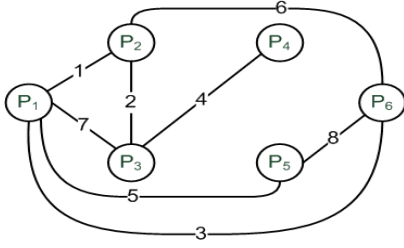


Fig. 2. An example of an input graph  $G_{\mathcal{I}}$  for the drug deal example.

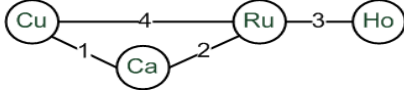


Fig. 3. The search graph  $G_{\mathcal{S}}$  for the specification of a drug deal scenario. The vertices of  $G_{\mathcal{S}}$  are labeled with the various roles of the persons involved in the drug deal.

- 3) For event variables belonging to two chains add an edge connecting their dependent entity variables.
- 4) Add two vertices connected with an edge for every event variable not appearing in any chain, representing its two free entity variables.

By construction, this graph will capture all the chaining constraints of the specification. To capture the temporal aspect of the behavior specification, we label each edge with the sequence number of the corresponding event variable in the search sequence  $\mathcal{S}$ .

Given these definitions, we can extend the notion of subgraph matching to the labeled graphs  $G_{\mathcal{S}}$  and  $G_{\mathcal{I}}$  as follows:

Find all the subgraphs of  $G_{\mathcal{I}}$  that are isomorphic to  $G_{\mathcal{S}}$  and that preserve the order of its labels.

The answer to the subgraph isomorphism will be the one-to-one correspondence  $f$  that maps every vertex of  $G_{\mathcal{S}}$  (equivalently every entity in our specification) to a vertex in a subgraph  $G'_{\mathcal{I}}$  of the input graph  $G_{\mathcal{I}}$  (equivalently entity in our input sequence). Preserving the order of the labels means that if we replace every vertex in  $G'_{\mathcal{I}}$  with its correspondent vertex in  $G_{\mathcal{S}}$  and we replace the labels of the resulting edges with their corresponding order we will end up with  $G_{\mathcal{S}}$ . Apparently, the proper assignment of labels to our search sequence, will be given by  $f$ .

As a simple example, consider the scenario described in Figure 1 and assume that we have the input graph presented in Figure 2. The pattern we wish to discover can be expressed intuitively in the form

$$(Cu, Ca), (Ca, Ru), (Ru, Ho), (Ru, Cu),$$

where apparently  $Cu(stomer)$ ,  $Ca(shier)$ ,  $Ru(nner)$ , and  $Ho(lder)$  are entity variables that can take values from  $\{P_1, \dots, P_6\}$ . We call such a pattern a *behavior specification*. A formal specification of the drug deal will be  $\mathcal{S} = x_1, x_2, x_3, x_4$  and  $C = \{x_1 \sim x_2 \sim x_3, x_1 \sim x_4, x_3 \sim x_4\}$ . The corresponding search graph appears in Figure 3.

It is easy to observe that  $G_{\mathcal{I}}$  contains the subgraphs  $G'_{\mathcal{I}_1}$  to  $G'_{\mathcal{I}_6}$  that appear in Figures 4.a- 4.f and are isomorphic to

$G_{\mathcal{S}}$ . However, only  $G'_{\mathcal{I}_4}$  (Figure 4.d) preserves the order of the labels of  $G_{\mathcal{S}}$ . The correspondence  $f$  for this graph will be  $f(Cu) = P_1$ ,  $f(Ca) = P_2$ ,  $f(Ru) = P_3$  and  $f(Ho) = P_4$ , and it will indicate the individuals matching the roles of our scenario.

### B. Proof Of Correctness

Given a subgraph  $G'_{\mathcal{I}}$  of  $G_{\mathcal{I}}$  that is isomorphic to  $G_{\mathcal{S}}$  it is easy to show that it will satisfy all the constraints of our specification and, thus, will be a matching pattern. In particular, there are three ways that  $G'_{\mathcal{I}}$  cannot be matching our specification. The first is that it can have more or less pair-wise events than our specification, but this would result in more or less vertices (when at least one of the entity variables in the event variables is free) or edges (when both entity variables in the event variable are dependent). The second is that there are pair-wise events that appear out of order, but this would mean that  $G'_{\mathcal{I}}$  doesn't preserve the order of the labels. Finally, the third is that some chaining constraint of the specification is not satisfied. The only two ways that this can happen is by a pair-wise event missing from the chain, say  $PE_i = (E_1, E_2)$ , or by two pair-wise events having two common entities. In the first case this would correspond in a missing edge in  $G'_{\mathcal{I}}$ , namely the edge between  $f(E_1)$  and  $f(E_2)$  and in the second case in a loop in  $G'_{\mathcal{I}}$ . However, in both cases  $G'_{\mathcal{I}}$  cannot be isomorphic to  $G_{\mathcal{S}}$ .

Similarly, assume that there is a matching pattern  $M$  in our input sequence, whose subgraph  $G_M$  (consisting of all pair-wise events in  $M$ ) on  $G_{\mathcal{I}}$  is not a valid isomorphic subgraph of  $G_{\mathcal{S}}$  that preserves the order of the labels. This means one of the following:

- 1)  $G_M$  and  $G_{\mathcal{S}}$  are isomorphic but  $G_M$  doesn't preserve the order of the labels. This however means that a pair-wise event occurred before some other event that preceded it in the specification, thus  $M$  cannot be a valid match.
- 2)  $G_M$  and  $G_{\mathcal{S}}$  don't have the same number of vertices or edges, which apparently can't be true or no one-to-one correspondence  $f$  from the vertices of  $G_M$  to the vertices of  $G_{\mathcal{S}}$  exists.
- 3) No one-to-one correspondence from the vertices of  $G_M$  to  $G_{\mathcal{S}}$  exists. In this case the degree sequences (sequences of the degrees of the graph's vertices in decreasing order) cannot be the same. However, vertices with degree more than one in  $G_{\mathcal{S}}$  correspond to dependent entity variables (it is easy to verify that vertices with degree one will correspond to free entities) and their degree will indicate the number of event variables of the chain in which they belong. Thus, the existence of at least a vertex with less degree in  $G_M$  than in  $G_{\mathcal{S}}$  (since both graphs have the same number of edges) will indicate that one of the chains is not satisfied and thus  $M$  cannot be a valid match.

### C. Subgraph Matching Algorithms & Complexity

The general problem of subgraph matching is known to be NP-complete (for general information on graph matching see

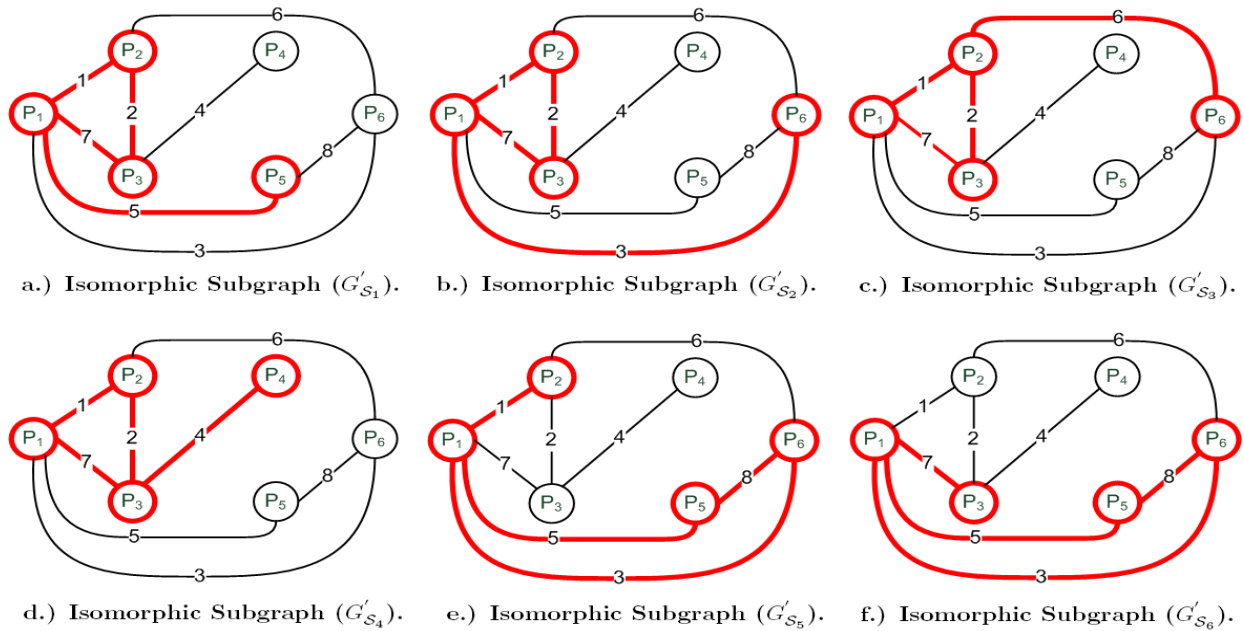


Fig. 4. Subgraphs of  $G_I$  isomorphic to  $G_S$  for the drug deal example. Figure (d) contains the detected drug deal.

[3]). However, it has been shown for a number of classes of graphs that the complexity of the problem is polynomial. One of these cases is for ordered graphs, which are graphs with the property that the edges adjacent to a vertex have a unique order [4]. In our case both  $G_I$  and  $G_S$  are ordered graphs, and thus the problem of finding all the copies of  $G_S$  in  $G_I$  has complexity  $O(N \cdot N')$ , where  $N$ ,  $N'$  is the number of edges in  $G_I$  and  $G_S$  respectively. The algorithm presented in [4] proceeds by creating a proper coding for the two graphs and traversing the edges of the graph with a particular order. Even for the case of general graphs efficient algorithms do exist. The VF2 algorithm [5], is based on backtracking and uses several rules to significantly prune the search space of the algorithm.

Moreover, integrating the requirement for a particular ordering of the labels is straightforward for algorithms based on backtracking as VF2. For a comparison of five of the most well-known general purpose subgraph matching algorithms the reader can refer to [6]. Incremental parsing for online subgraph matching is also possible. In the simplest case, we can only search for isomorphism the new subgraphs that are created by the addition of an edge.

#### D. Limitations & Extensions

Using the formal specification and the solution we presented, there is a number of patterns that we cannot search for. In particular, we cannot search for *multiple events between entities*, *events happening out of order* (i.e., events whose relative order is not important), as well as *events for which we know one or both of the entities involved*. Such specifications can be handled by allowing multiple labels, reusing labels in different edges and integrating vertex labels in our definition of subgraph isomorphism respectively. Note, however, that in

most cases this will come at the cost of increased complexity of the subgraph matching algorithm.

#### V. BACKGROUND INTERFERENCE

*Background interference* is defined as any event in our input sequence that is not a member of the pattern that we are trying to detect. The main side-effect of background interference is the creation of many “false alarms”. A *false alarm* is a legitimate instance of our pattern in the input sequence that is an artifact of background noise. Consequently, background noise can significantly limit the practical value of our algorithm by creating a large number of patterns for the human operators to verify.

Intuitively, the simpler the specification of a behavior, the easier it is going to be for a legitimate instance of the behavior to be generated randomly due to background noise. Moreover, the amount of background noise will depend on the number of entities in the monitored scene and on the number of pairwise events involving them. In a crowded scene, where only a few events occur, the generation of a sequence of events that match our specification will be rare. On the other hand, in a scene with just a few entities constantly interacting with each other, the generation of a simple sequence of events will have a high probability.

We will attempt to characterize the effect of background noise on the number of false alarms as a function of the number of different entities  $E$  and the number of pair-wise events  $N$  of the input sequence, as well as the complexity of our specification, which is given by the number of different entity variables  $E'$  and the number of event variables  $N'$  contained in the specification. Let  $\mathcal{F}_S$  be a random variable counting the number of false alarms given our specification. Ideally its expected value should drop fast as  $E$ ,  $E'$  and  $N'$

increase or  $N$  decreases. Assuming that the pair-wise events in the monitored scene are random, then our input graph  $G_{\mathcal{I}}$  can be modeled by a random graph. For reasons, of simplicity we consider that our input graph follows the  $G_{n,p}$  model, which means that  $G_{\mathcal{I}}$  is a graph with  $n = N$  vertices and edges that are selected with probability  $p = \frac{2N}{E(E-1)}$ <sup>1</sup>. It has been proven by Erdős (see [7]) that the expected number of subgraphs of  $G_{\mathcal{I}}$  that will be isomorphic to  $G_{\mathcal{S}}$  (which is a fixed graph), is given by

$$E[\mathcal{F}_{\mathcal{S}}] = \kappa p^{N'} \quad (1)$$

where  $\kappa$  is the number of copies (isomorphic subgraphs) of  $G_{\mathcal{S}}$  in  $K_E$ , which is the complete graph of order  $E$ . Apparently  $\kappa$  cannot be larger than  $\binom{E}{E'}$ , which is the number of different ways we can select  $E'$  of the  $E$  vertices of  $K_E$ . Thus, the expected number of false alarms will be given by

$$E[\mathcal{F}_{\mathcal{S}}] = \binom{E}{E'} \left( \frac{2N}{E(E-1)} \right)^{N'} \quad (2)$$

Note that since we don't take into account the ordering of the labels, *this expected value will actually be an overestimate*, especially as  $E'$  increases. This will happen because only a limited number of subgraphs of  $K_E$  will preserve the proper ordering of the labels. In particular, it is easy to observe that the probability of getting a sorted sequence of  $N'$  elements picked out of the set of natural numbers less or equal to  $N$  will be equal to  $\frac{\binom{N'}{N'}}{\binom{N'}{N'} \cdot N!} = \frac{1}{N!}$ , which apparently goes very fast to 0 as  $N'$  increases. Thus, even for busy scenes that people don't interact constantly with each other, the number of false alarms will be very small even for simple specifications. For an upper bound on  $E[\mathcal{F}_{\mathcal{S}}]$  the interested reader can see [8]. However, this bound is not very indicative for the behavior of our solution.

## VI. EVALUATION

In section 5 we provided a theoretical estimation of the expected value of false alarms as a function of the number of entities ( $E$ ) in our input sequence, the number of pair-wise events ( $N$ ) and the complexity of our specification expressed in terms of the number of entity ( $E'$ ) and event ( $N'$ ) variables it contains. In this section we will verify experimentally the performance of our proposed solution using two increasingly complicated scenarios of the drug deal example we introduced. The first scenario is the simple drug deal transaction model that we have used throughout the paper which consists of 4 event variables and 4 entity variables (see Figure 1). In a more elaborate scenario, we will use an extended drug deal

<sup>1</sup>Note that the  $G_{n,p}$  model might lead in graphs with more or less than  $N$  edges. In our case the random graph model that would better fit is the  $G_{n,M}$  which selects one of the possible  $\binom{n}{M}$  graphs with equal probability, and is essentially a snapshot of a random graph process  $\tilde{G}_n$ , which starts with  $n$  vertices and no edges and at each step adds one new edge. However, these models are closely related to the  $G_{n,p}$  (although proofs using the  $G_{n,M}$  and the  $\tilde{G}_n$  models are more difficult) and all the results that can be obtained for  $G_{n,p}$  hold also for  $G_{n,M}$  [7].

transaction model involving 6 entities and 7 pair-wise events. This scenario consists of the following steps:

- 1) Negotiator approaches Cashier (make a deal).
- 2) Negotiator signals Banker (everything ok).
- 3) Banker approaches Cashier (brings the cash).
- 4) Cashier signals Runner (cash received).
- 5) Runner picks up drugs from Holder (get drugs from safe place).
- 6) Runner delivers drugs to Carrier (deliver drugs).
- 7) Carrier signals Negotiator (drugs delivered).

### A. Simulation Design

The main tool for evaluating our work in group behaviors is the Virtual Battle Space 2 simulator from Bohemia Interactive. VBS2 is based on a game engine mainly used for military simulations. VBS2 can simulate a large number of interacting entities providing detailed information for the motion, posture and activity of all the entities (people, cars, objects) in the monitored scene. Using VBS2 we can create playable scenarios, involving large number of human-operated characters and we can have very intelligent entities interacting in the background. In an attempt to evaluate the proposed solution, we implemented the simple drug deal example of Figure 1 in VBS2<sup>2</sup>. Figure 5 shows the trajectories of the people in the monitored scene, in terms of location and duration spent at a location. A close examination of this plot and the video shows that the interactions between individuals can be detected using physical proximity and a time interval. Because of the limited number of interactions in this simulation, it is easy to detect the subgraph corresponding to the drug deal.

To generate much larger scenarios that what can be rendered in VBS2 we have also implemented a lightweight simulator in Python, that can mimic the interactions in bulk. The simulator generates a sequence of pair-wise events which are noisy in that satisfying subsequences, with respect to a given behavior specification, may be interleaved with pair-wise events which are unrelated to the behavior of interest. This can also capture partially the effect of sensing noise.

### B. Experimental Results for the Effect of Background Interference

Attempting to verify experimentally the effect of background noise, Figure 6 presents the average number of false alarms in our input sequence for the case of the simple drug deal and for different number of entities and pair-wise events. The average number and the standard deviation have been evaluated for a large number of simulation traces and different values of our parameters and the reported results for every selection of parameters involve at least 20 traces. In all our plots you can see that the standard deviation (which appears as error bars) is relatively small and, hence, the reported average value is a good indication of the expected behavior of our algorithm.

<sup>2</sup>An annotated video of one of the simulations can be found at <http://www.youtube.com/watch?v=eFPEXg1XAaU>. Note, however, that in this rendering there is no interaction (i.e., physical proximity) between the cashier and the runner.

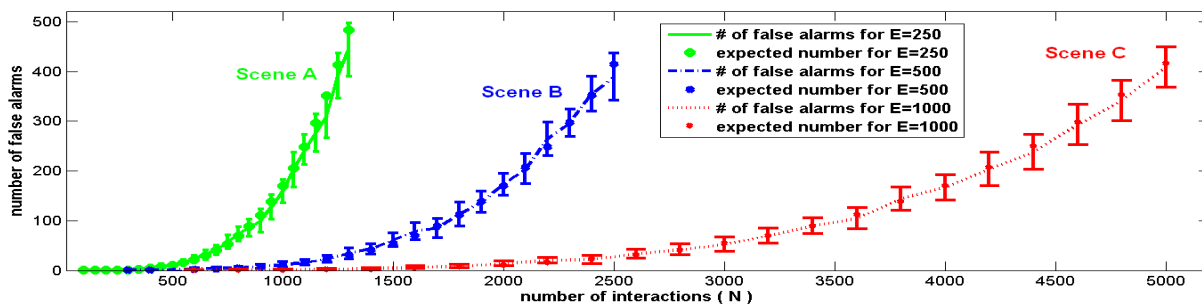


Fig. 6. Average number of false alarms for the simple drug deal scenario using different values of number of events in our input sequence ( $N$ ) and different number of entities ( $E$ ). Standard deviation appears as error bars. In the same plot appears also the theoretically expected value of false alarms given by equation (3) as estimated in section 5.

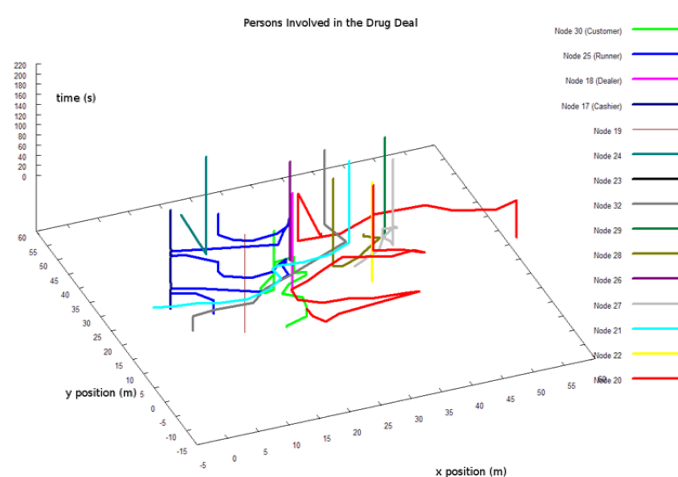


Fig. 5. The trajectories of the persons in the scene of a simulation of the simple drug deal using VBS2.

Concerning the correctness of our algorithm, all our traces contain at least one drug deal starting at a random point in time, which our algorithm was able to extract in every case. This, however, is anticipated since our Python simulator cannot currently reproduce sensing noise, which would give ambiguity for the presence of some interactions. Part of our plans for future work includes extending our algorithm and problem formulation to the case where there are ambiguities in the sensed data. We note also that in the scenarios we perform our evaluations on, the entities involved in the drug deals are also allowed to interact with “background characters” i.e. individuals in the scene who are unrelated to the drug deal. This increases the probability for false alarms since each of the protagonists of the drug deal will already have partial involvement in some of the interactions required by the specification.

Concerning the number of false alarms, the first curve (Scene A) corresponds to a scene with 250 monitored entities (persons) interacting with each other. For this scene we can see that the number of false alarms will tend to 0 for less than 500 interactions. For the other two monitored scenes, namely Scene B with 500 entities and Scene C with 1000 entities the respective numbers of interactions for 0 false alarms are

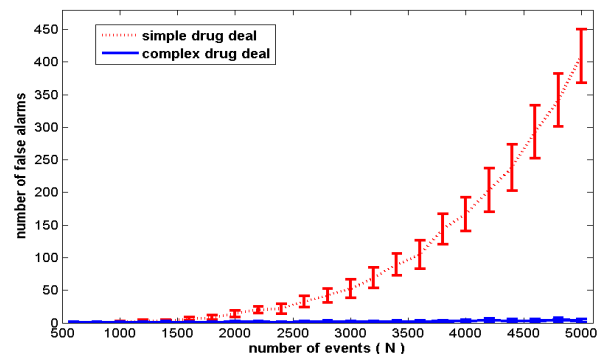


Fig. 7. Comparison of the average number of false alarms for the 2 scenarios, number of entities  $E = 1000$  and different values of  $N$ . Standard deviation appears as error bars.

1000 and 1700 respectively. If you consider Scene C as a train station terminal populated with a 1000 travelers (we assume it is easy to separate travelers from employees), what this means is that travelers in the train station terminal can have on average at least 1.7 interactions with each other before we can get a single false alarm for a simple specification containing 4 entity and 4 event variables. Moreover, you can see that the number of false alarms will be relatively low for significantly large numbers of pair-wise events. For instance the surveillance camera operators of the train station in Scene C will have to check less than 100 instances of suspicious behavior in a scene involving 1000 individuals and 3600 interactions among them (3.6 on average per person).

Another interesting observation is that the steepness of the curves decreases as the number of entities in the monitored scene increases. This observation is anticipated by the analysis of section V, and implies that in more crowded scenes we can have a larger number of interactions before false alarms start to occur. Again this is very important for the type of applications in which we are interested, where we assume that we will have a large number of monitored entities.

For the case of the more elaborate drug deal scenario, the number of false alarms for 1000 entities appears in Figure 7. For comparison, we provide in the same plot the number of false alarms for the case of the simple scenario. From this comparison it is easy to infer the effect of the elaboration of the scenario in the number of false alarms, since for even

5000 interactions (5 on average per person) the number of false alarms will be approximately 0. What this means is that *the more elaborate the behaviors of interest are and, thus, the more difficult for human operators to notice, the more accurate our solution will be.* Another important implication of this characteristic of our solution is that *starting from a very vague description of a behavior consisting from just a subset of its events that are known, we can detect it (along with many similar activities) and then with the help of a human create a much more detailed specification that will match only the behavior of interest.*

Finally, concerning the expected values from the analysis of section V (values given by equation (2)) these appear, for the case of the simple scenario, as points on the corresponding curves of Figure 6. From these points, we can see that for the case of our simple scenario, where  $E' = 4$  and  $N' = 4$ , the theoretically expected value matches very closely the average numbers of the simulation, something which also verifies partially the accuracy of our simulation model. In the case of the extended scenario ( $E' = 6, N' = 7$ ) the expected values from equation (2) are actually significantly higher than the actual number of false alarms in our simulation. For example for the case of 5000 interactions ( $N = 5000$ ) the expected number of false alarms is 13.8, whereas in our simulations the corresponding number of false alarms was 4.3 (i.e., almost 3 times less).

## VII. RELATED WORK

The literature on group behavior detection can be divided into two groups. The first deals with the case where it is not known beforehand which behaviors or entities are of interest, and data mining approaches such as clustering are applied to discover “similar” entities, “normal” behaviors and “abnormal” behaviors [9]. The second class of behavior detection problems assumes that the behavior of interest is either known or partially known, as is the case for the problem considered in this work. In [10], [11] for example, language based formalizations are proposed for describing behaviors of interest among entities of known “types,” i.e. airplane, person or truck. The approach used in this work for detecting a group activity specified through a sequence of event variables can roughly be characterized as one of mining on dynamic graphs, i.e. graphs with time-varying edge (vertex) weights.

Generally speaking, frequent subgraph mining or subgraph discovery on static graphs is a well studied problem; we refer the reader to [12] for a survey of various mining approaches on static graphs. However, the temporal aspect of behaviors involving multiple entities is not considered in such problems. The problem we consider in this work also bears some superficial resemblance to the sequence matching problem where the objective is to determine if a specified subsequence exists in a given sequence (e.g., see [13], [14]). However, a number of differences make the solutions and approaches of sequence matching, and its various extensions, inapplicable to the problem at hand. The key difference is that the specification describing the behavior we aim to detect consists of event variables and constraints among the event

variables. In this context, the solutions and approaches to the sequence matching problem do not in general apply.

## VIII. CONCLUSIONS & FUTURE WORK

In this paper we described a method for detecting chains of events taking place under the field of different cameras and for extended periods of time given a simple specification and assuming no knowledge about the existence of groups or the identities of the entities involved. We presented a solution based on converting the problem to an equivalent subgraph matching problem and we studied its properties with respect to interference from other interactions taking place in the background. Despite its complexity the proposed algorithm was shown to be useful in practical settings.

In our future work we plan to examine the cases where the human operators do not know the exact definition for the chain of events. For this we will investigate ways of isolating chains of events of interest based on a looser definition. At the same time, we are already investigating methods for extracting frequently recurring patterns and patterns with some regularity that will assist the human operator to provide the system with more accurate definitions of what chains of events need to be detected from the data.

## REFERENCES

- [1] K. Bernardin, T. Gehrig, and R. Stiefelwagen, “Multi- and single view multiperson tracking for smart room environments,” in *CLEAR*, 2006, pp. 81–92.
- [2] O. Javed, Z. Rasheed, O. Alatas, and M. Shah, “KNIGHT: A real time surveillance system for multiple overlapping and non-overlapping cameras,” in *IEEE International Conference on Multimedia Expo*, 2003.
- [3] H. Bunke, “Graph matching: Theoretical foundations, algorithms, and applications,” in *Proc. Vision Interface*, 2000, pp. 82–88.
- [4] X. Jiang and H. Bunke, “Marked subgraph isomorphism of ordered graphs,” in *SSPR '98/SPR '98: Proceedings of the Joint IAPR International Workshops on Advances in Pattern Recognition*. London, UK: Springer-Verlag, 1998, pp. 122–131.
- [5] L. Cordella, P. Foggia, C. Sansone, and M. Vento, “A (Sub) Graph Isomorphism Algorithm for Matching Large Graphs,” *IEEE Transactions On Pattern Analysis And Machine Intelligence*, pp. 1367–1372, 2004.
- [6] P. Foggia, C. Sansone, and M. Vento, “A performance comparison of five algorithms for graph isomorphism,” in *Proc. of the 3rd IAPR TC-15 Workshop on Graph-based Representations in Pattern Recognition*, 2001, pp. 188–199.
- [7] B. Bollobás, *Modern Graph Theory*. Springer, 1998.
- [8] S. Janson, K. Oleszkiewicz, and A. Ruciński, “Upper tails for subgraph counts in random graphs,” *Israel Journal of Mathematics*, vol. 142, no. 1, pp. 61–92, 2004.
- [9] J. L. Patino, E. Corvee, F. Bremond, and M. Thonnat, “Data mining for activity extraction in video data,” in *EGC 2008*, 2008, pp. 433–444.
- [10] Fusier, Florent, Valentin, Valery, Bremond, Francois, Thonnat, Monique, Borg, Mark, Thirde, David, Ferryman, and James, “Video understanding for complex activity recognition,” *Machine Vision and Applications*, vol. 18, no. 3-4, pp. 167–188, August 2007. [Online]. Available: <http://dx.doi.org/10.1007/s00138-006-0054-y>
- [11] A. Hakeem and M. Shah, “Learning, detection and representation of multi-agent events in videos,” *Artif. Intell.*, vol. 171, no. 8-9, pp. 586–605, 2007.
- [12] L. Getoor and C. P. Diehl, “Link mining: a survey,” *SIGKDD Explor. Newsl.*, vol. 7, no. 2, pp. 3–12, 2005.
- [13] M. Ruotsalainen, T. Ala-Kleemola, and A. Visa, “Gais: A method for detecting interleaved sequential patterns from imperfect data,” *Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on*, pp. 530–534, 1 2007-April 5 2007.
- [14] K. M. Risvik, “Approximate word sequence matching over sparse suffix trees,” in *CPM '98: Proceedings of the 9th Annual Symposium on Combinatorial Pattern Matching*. London, UK: Springer-Verlag, 1998, pp. 65–79.