Understanding Interactive Dynamic Situations

Jérôme Thoméré, Simon King, Sophie Motet, François Arlabosse
Framentec-Cognitech
Tour Fiat - 92084 Paris La Defense - FRANCE
+33 (1) 47 96 46 62
email : jthomere@framentec.fr

Abstract

The system presented here is developed to interpret dynamic situations involving several moving objects in a known environment. It is part of the ESPRIT VIEWS project and processes data coming from sequences of video frames. These numerical data are interpreted to identify in real-time situations including complex interactions between moving objects in the scene. The principle is to compile pre-defined behaviours for individual objects then for groups of objects from elementary events. The method used to accomplish this is based on temporal networks and dynamic grouping. The applications studied here consists in the detection of incidents on an urban roundabout.

1 Introduction

The issue is to interpret situations involving several moving objects in a known scene. This can be used for detecting particular situations (incidents in road traffic, surveillance of sensitive zones...) or for off-line requirements such as statistics, planning checking...

The features of the scene observed should meet a certain number of conditions :

- The scene is known a priori. In particular, the spatial layout can be decomposed into meaningful regions.
- The objects are of known types.
- Information is given sequentially, each update giving the position, velocities or any other relevant data about each object.

1.1 State of the art

Very little work has already been done on such subjects. Most of them are related either to very constrained and deterministic relationships or to individual behaviours.

In Karlsruhe University a similar approach has been followed by Nagel's work[9]. However as far as the conceptual part is concerned, they are interested only in individual behaviours especially regarding the completion of scripts (Nagel gives the example of a gas-station scenario). Moreover, their linguistic approach gives a more rigid use of time than in our work.

What are the essential problems posed by these kind of applications ? Mainly they are of two types :

- Individual behaviours and *a fortiori* interactions between behaviours are not predictable from a long term point of view.
- They can a priori occur between all objects (especially if we consider "massive" behaviours such as queuing) and it is not difficult to realise that this can make the problem untractable.

2 The method: GEM

The aim of the method chosen was to cut the recognition process into different layers according to the level of complexity of the items recognised. The general recognition process adopted the following scheme :

Frames
↓
Events
↓
Individual behaviours
↓
Interactive situations
(group behaviours)

To summarise each step, let us say that the first step is a very procedural process, based on geometrical and kinematical models ; the second step that transform EVENTS into BEHAVIOURS is based on propagation of temporal values in a constraint network. The last step is very similar to the second one, except that it includes

one of the keys to the overall method : the dynamic grouping of objects into significant groups.

This approach was originally based on the A. Lansky's GEM framework [7, 8]. The ideas kept were notably the event-based reasoning with temporal calculus and the notion of groups to reduce complexity.

2.1 Events

We assume we have a temporal sequence of snapshots of the world. In the application considered, these are the result of vision processes from video frames, but this is not compulsory. To transform such a sequence into individual behaviours, the idea is to compute events, which are considered to be instantaneous behaviours. As has been mentioned above, this idea comes from GEM, although it was originally intended for planning purposes. Lansky wanted to find an alternative approach to the usual state-oriented world models and claimed that it was more natural and efficient especially for the representation of concurrent actions.

One difference here with Lansky's work is that we are provided data which are represented in a basically state-based way, since each element of the sequence (here frames) gives information such as position, class, velocity... of the objects seen in the scene.

To identify events from such data, the approach followed here is totally procedural. We have identified three main types of events for each of which we will detail the method used, but this may depend on the application.

- **Kinematical events** are events related to change of velocity vector, either in direction or in speed (start, stop, accelerate, decelerate, turn-right, turn-left). They are calculated simply by comparing the velocity of the object in the current frame and in the previous one. For instance ACCELERATE is computed with the model

$$||\vec{v}_2|| - ||\vec{v}_1|| > \theta$$

  where θ is a threshold which may depend, for instance, on the class of the object.

- **Spatial events** are events related to the spatial environment (enter-region, exit-region). To compute them, the spatial layout has been decomposed into significant regions (see fig 1). Regions and segments of lines that separate regions are stored in a spatial database . For each object the position of the two last frames are extracted. If this last segment crosses a boundary segment events ENTER-REGION and EXIT-REGION are identified. Since regions can be grouped into meaningful bigger regions (e.g. a GIVE-WAY REGION), these elementary events can lead to other spatial events such as ENTER-GIVEWAY-REGION.

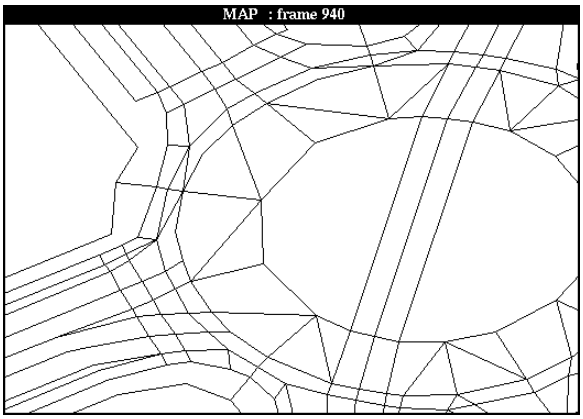


Figure 1: *Map of the scene*

- **Relational events** are events associated to a given object relative to other vehicles. They are considered as individual events in so far as the other vehicle is here just considered as a part of the outer world. In our application we have considered two sort of relational events, according to the mode of calculus.

  – *Kinematical.*

  For example the FOLLOW event model is

$$\frac{|\vec{v_1}.\vec{P_1P_2}|}{||\vec{v_1}||.||\vec{P_1P_2}||} > cos(\frac{\pi}{2})$$

  – *Analogical*

  This type of event is recognised via a tessellation of the spatial layout into cells[10]. They use the "path" drawn by a vehicle when it moves. An event such as FOLLOW-PATH means that a vehicle is moving in the path of another vehicle i.e. it is where the other one was in the past. Such events are very useful in queuing situations.

2.2 Temporal calculus

In order to build behaviours on events and other behaviours, we have to relate them. These relations are mostly temporal and this idea was also presented by Lansky in [7, 8]. This is why we build a network with temporal operators as nodes. These temporal operators

represent laws of internal composition in an interval algebra. This last point is the main difference with usual temporal logics such as Allen's[1, 2] but we are going to develop this point later. We tried to use the TMM of Dean[5, 4] but it appeared to be a bit too heavy and powerful for what we needed[6, p7–24].

From the above events, we want to deduce behaviours. For instance we wanted to deduce the behaviour GIVE-WAY from the sequence of events ENTER-GIVEWAY-ZONE → DECELERATE → STOP. But we also need to infer behaviours from other behaviours. The operations that will allow these inferences are temporal. That is to say we are going to build models such as

BEHAVIOUR = EVENT1 OP EVENT2

or

BEHAVIOUR = BEHAVIOUR1 OP' BEHAVIOUR2

where OP and OP' are temporal operators.

The semantics attached to a behaviour or an event are temporal intervals. Therefore the interpretation of OP's are operators of laws of internal composition in an interval algebra. This is really similar to Allen's interval logic. The only difference is that when we compare two intervals, we do not only want to know if they satisfy the relation, but we also want to know the temporal interval during which this relation is satisfied. In other terms, the composition of two temporal intervals must return an interval and not a boolean. This implies that the relations are potentially far more than thirteen. However we did not see the need to implement all the possible operators since most of the virtual temporal laws would be absolutely useless. In practical applications we create the operators we know we could need.

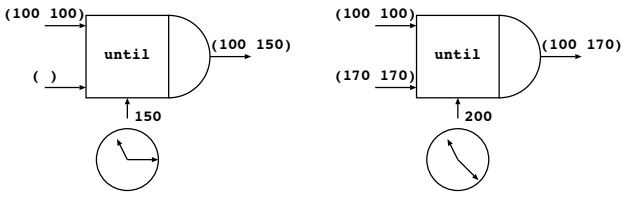


Figure 2: *Example of the* UNTIL *operator*

Moreover, this algebra has also to be extended to take the actual time into account. We need for instance the operator UNTIL which means that a behaviour is true as soon as an EVENT1 occurs and ceases when an EVENT2 occurs. However, as we want a continuous result, we have to give a value to the result : the end of the interval returned is given by an internal clock representing the current time. In general the CLOCK is used whenever we want to reason on default assumptions : the absence of an event/behaviour must be stamped.

2.3 Propagation in networks

To operate this temporal algebra we have chosen to use a network approach. The networks we build are therefore made of temporal operators connecting events/behaviours together. They are the internal representation of the above models. A network is attached to one particular object (element or group) and compiles all the behaviour models that are associated to this object. As we will see later, networks can be connected to one another.

Without going into the details of the implementation,in addition to nodes corresponding to temporal operators (called PROCESSORS) there are nodes which are used to store values propagated.

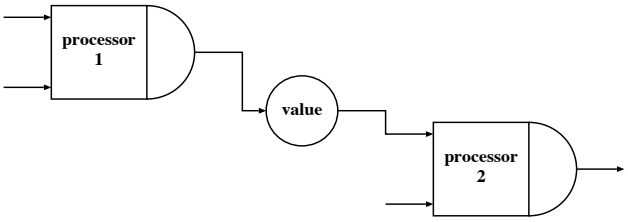


Figure 3: *Nodes of the network*

Temporal values (intervals) are propagated along the arcs. Each processor has a defined set of terminals with particular roles. On the other hand each value node may be connected to as many other nodes as needed. Each time a value comes through one connection of the nodes, it is instantaneously propagated to the other ones.

The values may be more complicated than simple intervals. This occurs especially for *attributed* events/behaviours such as ENTER-REGION (in this case the attribute is the name of the region). Here the values propagated are sets of pairs <attribute, interval>. We can have values such as ((r1 100 200) (r2 210 240) (r3 250 310)). This does not change the fundamental mechanism.

Internal rules are defined for each type of processor node. It defines :

- Its triggering modes, i.e. which sets of its terminals will trigger the computation.
- For each mode, the result of composition of inputs.
- For each mode, the terminal which will propagate the result.

Contrary to what the figures may let us believe, there is no absolute distinction between input nodes and output nodes. The propagation of values could easily follow any direction. This is a means for handling incompleteness. For instance, if we know that BEHAVIOUR1

is the result of composition of BEHAVIOUR2 and BEHAVIOUR3, that BEHAVIOUR3 occurred, and also that BEHAVIOUR1 occurred, we could deduce the occurrence of BEHAVIOUR2 in spite of the absence of data on it.

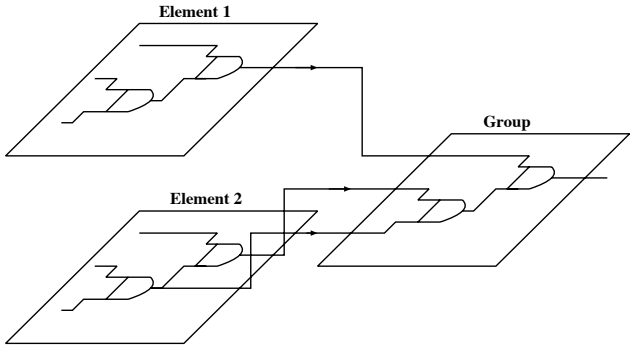


Figure 4: *Connection of networks*

2.4 Group behaviours

The above description is valid for network attached to one particular object, but we can define interactions between individual behaviours as behaviours of groups of individuals. A simple example of such a group behaviour could be the BLOCKING behaviour, when a vehicle stops in front of another one, forcing it to stop. It is obvious that we could not attach such a behaviour to anyone of the two objects without considering at least events attached to the other one. We need to consider behaviours attached to both. Of course group behaviours will be computed from individual behaviours (here, STOPPING for one object, FOLLOWING and STOPPING for the other one). In this case a natural group to be defined is a group of two objects following each other.

So this means we connect individual networks to group networks. How shall we make this connection ? Are we going to connect every element to every other element ? This would no doubt lead to combinatorial explosion.For instance, if you considered all possible sets of vehicles to test if they are queuing you would have to examine 2^n possibilities. In order to make the problem tractable, we have to group a priori elements according to their plausible relationships. That is the reason of grouping.

2.5 Dynamic grouping

The idea of grouping comes also from GEM. Yet, in [7, 8], the groups envisaged are static since they are *a priori* defined.

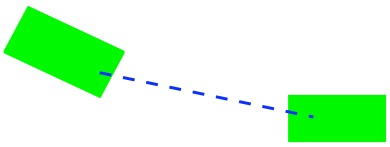
As objects in the scene evolve with time (their state changes, they appear, they disappear...), the grouping must be dynamic. On which criteria shall this grouping be based ? They must depend on the type of group behaviour we want to recognise.

Three types of grouping are possible :

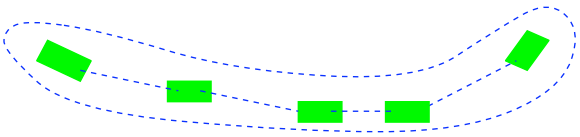
1. *Centered around an object*
2. *Based on relational events*
3. *Based on the static environment*

In the application we worked on, we have implemented the two last ones. We therefore have three types of groups :

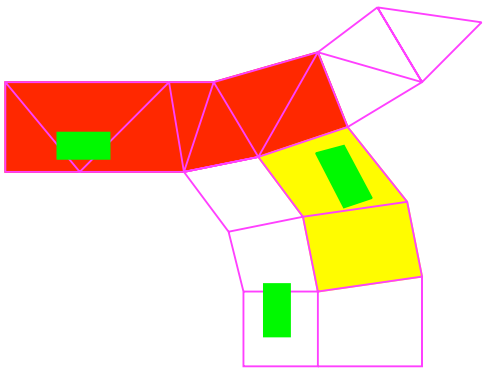
- **Binary groups** They are built with the event FOLLOW. Each time an object follows another one, they are grouped together into a binary group.



- **Queue groups** They are built from the previous type of groups. When two binary groups shares an element, they form a queue group.



- **Giveway groups** Some of the big regions of the spatial layout are labelled as GIVE-WAY REGIONS i.e. they are regions where the vehicles must give way. Some are labelled as PRIORITY REGIONS in



which vehicles are given way. Moreover regions are grouped in pair (GIVE-WAY to PRIORITY). Each pair will define a group. Of course the group will be existent only if it has some vehicles inside both regions.

There are three main gains with dynamic grouping :

- It reduces computational cost. This was the original goal of such a construction.
- It allows to define and implement group behaviour models.
- It generates new events. For example, the simple fact of creating a new QUEUE GROUP can be considered as a QUEUING event.

The main challenge is to make dynamic grouping truly real-time. The bottleneck is the passage from elements to groups.

3 The system : the VIEWS application

What we are describing in this paper is only a part of the overall VIEWS system. We are not going to describe the perceptual part which uses artificial vision based techniques.

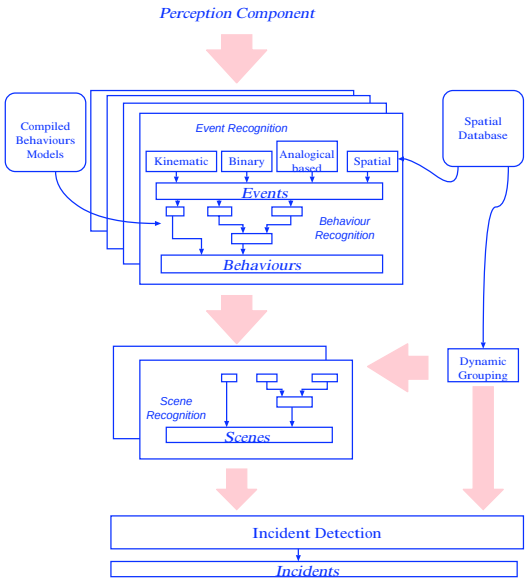


Figure 5: *Diagram of the conceptual module*

We can nevertheless summarise very quickly the process preceding the conceptual module :

Currently the processes run through video frames. A seeding subsystem based upon motion detection in the frames gives a first indication of the location and type of the vehicles present in the scene. A model matcher alerted by the former identifies the objects and their position in 3D.

This is where the conceptual module we describe in this paper intervenes. The updates given by the model matcher are for instance given every tenth frame. Each one gives the classification, position, bounding box, direction and velocity of each object recognised. They are provided in the following format :

```
(FRAME ID 0 OBJECTS (

(OBJECT ID 4 CLASS xxx SCENE_POS (x y z)
B_BOX (...))
(OBJECT ID 3 ...)
...)

(FRAME ID 10 OBJECTS (...))

...
```

The conceptual module follows the architecture described in Figure 5. For each object procedural event recognition is performed, then seeded in the individual network, then propagated in group networks (initially formed with dynamic grouping).

The conceptual module computes individual events, individual behaviours, group behaviours as described above and detects pre-defined incidents in the following list :

- Refusal of priority (fig6). A vehicle is exiting a give-way region (defined in the spatial database) when another is approaching and has slowed down.
- Formation of queues. Queues are defined as groups of 3 or more vehicles each following each other.
- Queue breaking up.
- Objects leaving queues.

To relate briefly how the results appear, let us say that during the session, we see vehicles moving on the map (see fig7). Events are detected one or two updates (about 1/2 seconds) after they occur. The dynamic groups are graphically displayed on the map, linking the objects together. When an incident is detected, the zone in which it occurred is highlighted and the incident

is briefly related in a special window. It is always possible to come back to a former incident to get detailed explanations or a replay of it. You equally have access to the temporal history of elements and groups.

Figure 6: *View of refusal of priority*

So far, without any optimisation of the code or of the algorithms implemented, the process is a bit slower than real-time. It takes less than five times real-time for around ten objects, six binary groups and one queue present in the scene. It is relatively easy to see that if the number of vehicles or groups is limited, all the processes related to them can be parallelised. As we have already said, the bottleneck is clearly the dynamic grouping. A control component is planned to achieve real-time.

4 Conclusion

What we have presented here is part of a vision system, but it should be apparent that this is not the only possible application. One could imagine a very similar version of this system obtaining its information from a radar,infrared,etc. It is even not compulsory for the system to deal with spatial positions, the relationships between elements might be completely different, such as the components of an electric installation...

Nevertheless, the most immediate application seems to be surveillance. All types of surveillance are potentially relevant to this system, but it should be easier in more constraint areas such as factories with robots. Otherwise to start a non exhaustive list, we have :

- Road traffic control.
- Aircraft servicing verification.
- Surveillance of sensitive areas.
- Military applications.

One compulsory improvement is to make the system truly real-time.As a first step, it can be parallelised on objects and groups. The second and more difficult step will be for dynamic grouping.

A complete system may use the information given by the conceptual module to control the perceptual module, regardless of its form. For instance if the conceptual module knows what the class of an element should be, it could inform the model-based module of this class.

The system should also be able to cope with incompleteness. In the application we are currently looking at, two interesting cases are envisaged : long-term occlusions and temporal reasoning with missing events.

All these improvements are currently being worked on and are part of what we have called the CONTROL COMPONENT.

5 Acknowledgements

This work was done principally at Framentec-Cognitech (France). We would like to thank Andrew Toal (Queen Mary and Westfield College - UK) for his valuable analogical contribution and all the partners involved in the VIEWS project (especially Marconi Radar and Control Systems) for fruitful debates.

References

[1] James F. Allen. An interval based representation of temporal knowledge. In *IJCAI-81*, pages 221–226. IJCAI, Morgan Kaufmann, 1981.

[2] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, November 1983.

[3] The ESPRIT P.2152 VIEWS Consortium. The views project and wide-area surveillance. Technical Report PM-02-ECCV92-01, The ESPRIT P.2152 VIEWS Consortium, April 1992.

[4] Thomas Dean. The tmm manual. Technical report, Brown University, 1987.

[5] Thomas L. Dean. *Temporal Imagery: An Approach to Reasoning about Time for Planning and Problem Solving.* PhD thesis, Yale University, Dept. of Computer Science., May 1986.

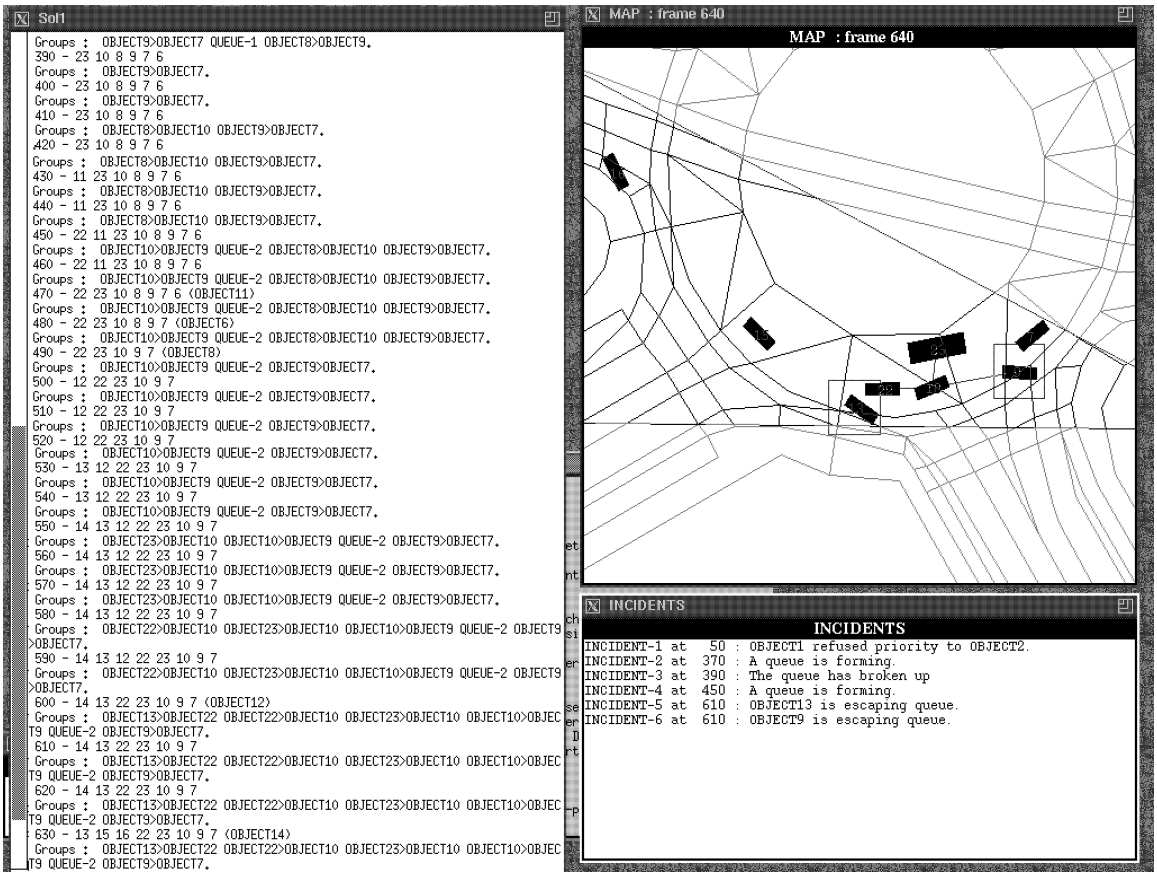


Figure 7: *Aspect of a VIEWS screen*

[6] Vu Duong, Hilary Buxton, Richard Howarth, Paddy Toal, Gong Shaogang, Simon King, John Hyde, and Jérôme Thoméré. Spatio temporal reasoning (i). Technical Report PM-03-CEC.D203, The ESPRIT P.2152 VIEWS Consortium, 1990.

[7] Amy L. Lansky. A representation of parallel activity based on events, structure, and causality. Technical Report 401, SRI International, AI Center, December 1986.

[8] Amy L. Lansky. Localized event-based reasonning for multi-agent domains. Technical Report 423, SRI International, AI Center, 1988.

[9] Hans-Helmut Nagel. The representation of situations and their recognition from image sequences. In *Congrès AFCET-RFIA, Lyon*, pages 1221–1229, November 1991.

[10] Andrew Toal. Spatio-temporal reasoning within a traffic surveillance system. In G. Sandini, editor, *ECCV 92*, pages 884–892. DIST, University of Genoa, Springer-Verlag, May 1992.