A Visual Surveillance System for Incident Detection

Simon King, Sophie Motet, Jérôme Thoméré & François Arlabosse
Framentec-Cognitech S.A.
Tour Fiat, Cedex 16,
92084 Paris La Defense,
France.

E-mail : sking@framentec.fr

Abstract

A conceptual module capable of scene interpretation for incident detection has been implemented using a relational network approach and a concept of dynamic grouping to ease the complexity involved in computing multiple object behaviours. This system has been tested on several roundabout scenarios and has given good results using simulated perfect data. Using more realistic data necessitated the handling of multiple hypotheses on the object classes. This was easily handled using the method developed and even though this increased the complexity of the system, on-line response was maintained.

1 Introduction

This paper describes some of the results achieved over the four years duration of the ESPRIT II P.2152 VIEWS[1] project and attempts to evaluate these against both end-user's expectations and more "technical" criteria. The results and evaluation presented will be on what was termed the *conceptual module* in the project, namely the transformation from numerical to symbolic data and the reasoning done with the latter in order to recognise so-called *incidents* in a road-traffic scenario. We shall start with a quick overview of the project and then describe our approach to solving the conceptual part of the system, based on a relational network approach. Given this we will present the results of our work.

2 The VIEWS Project

The VIEWS project is an ESPRIT II project involving six partners and lasting four years which were completed last March. Its goal was to build surveillance systems for wide-area scenes, with one of the demonstrators being targeted to road traffic incident detection (two other demonstrators focused on airport applications). The chosen road traffic scenario was of a roundabout characterised by having several entry and exit lanes and, in addition, a tram track crossing it. A representation of the scene can be seen in figure 1. It should be noted that in the project we limited ourselves to a single fixed camera, whose field of view did not cover the whole roundabout.

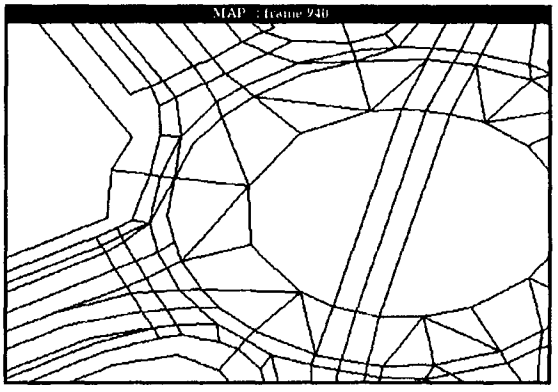


Figure 1: *Map of the scene*

As stated previously, the objective of the VIEWS project was to produce complete surveillance systems. As such the overall architecture can be seen in figure 2. We can see that there are three main components :

- **Perception Module** — This module takes in images coming from a standard video camera. It then detects motion, tracks coherently moving *blobs* and then a 3D model matcher attempts to identify the object (i.e. the class of the object). The output of this module is essentially, for each frame, a list of objects, their class, position in the scene and image, and orientation.

[1] VIEWS stands for Visual Inspection and Evaluation of Wide-area Scenes

- **Conceptual Module** — This module takes as input the numerical data provided by the perception component, transforms these into symbolic events and reasons with these in order to perform behaviour recognition for both single and multiple objects.
- **Control Module** — The role of this component is to ensure that we have an active vision system, with the above two components working towards the same goal. It is currently used to provide a focus of attention mechanism.

Additionally there are the spatial database that stores a representation of the scene that has been decomposed into various regions; and the behaviours database that contains the definitions of the behaviours to be recognised.

It is the conceptual module that we worked on and that is the main subject of evaluation. In order to do so we will now discuss in some detail our approach for behaviour and incident detection. Further details can be found in [1, 2].

3 The Conceptual Module

The conceptual modules takes as input the essentially numerical output from the perception component. This data is organised by frame and contains information on each object in the scene, their position in both the scene and the image, their orientation, their velocity and their class (i.e. the type of object, for example saloon) and a belief value associated with the latter. Given this data the following approach was adopted :

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

Our method, based on ideas from Lansky's GEM framework [3], to accomplish each of the above steps is now described.

3.1 Event Computation

The process here uses the raw output from the perception component and computes events in a procedural manner. The events that are calculated are those that were needed to compute our behaviours (and hence incidents) but nevertheless seem to be reasonably complete for most purposes. These events can be considered to be of three types :

- **Kinematical events :** these are events that are related to a change in the velocity vector, either in direction or in speed. Computation is straightforward by comparing an object's velocity in the current and previous frames. Typical events computed are START, STOP, ACCELERATE, DECELERATE, TURN-RIGHT, TURN-LEFT.
- **Spatial events :** these are events computed relative to the spatial layout. The scene considered has been decomposed into significant regions (see figure 1) which are stored in a spatial database (part of the dynamic database in figure 2) in terms of regions and line segments that separate regions. For each object the position of the last two frames are extracted and if the latter crosses a boundary segment the events ENTER-REGION and EXIT-REGION are recognised. We should also note here that regions can and are grouped into meaningful larger regions (such as GIVE-WAY-ZONE) which leads to other events. We will see later the importance of these larger regions.
- **Relational events :** these are events associated with a given object relative to other objects. They are however considered as individual events in so far as the other object is just considered as a part of the outside world. Two sorts of relational events are computed based on different methods. These are :
  - *Kinematical*
    For example the FOLLOW event model is
    $$\frac{|\vec{v_1}.\vec{P_1P_2}|}{||\vec{v_1}||.||\vec{P_1P_2}||} > cos(\frac{\pi}{6})$$
  - *Analogical*
    This uses the path drawn out by an object as it moves thanks to a tessalation of the spatial layout into cells. The FOLLOW-PATH event is thus recognised when an object moves in the path of another object (i.e. where it was previously in time).

These are the events that are calculated for each object for each frame. It should be noted here that we only sample the perception component output every 10th frame — given that the video output rate is 25 fps, this corresponds to approximately every 1/2 second. This is more than satisfactory for urban road traffic scenarios. Now that events are recognised we can proceed to single object behaviour recognition.

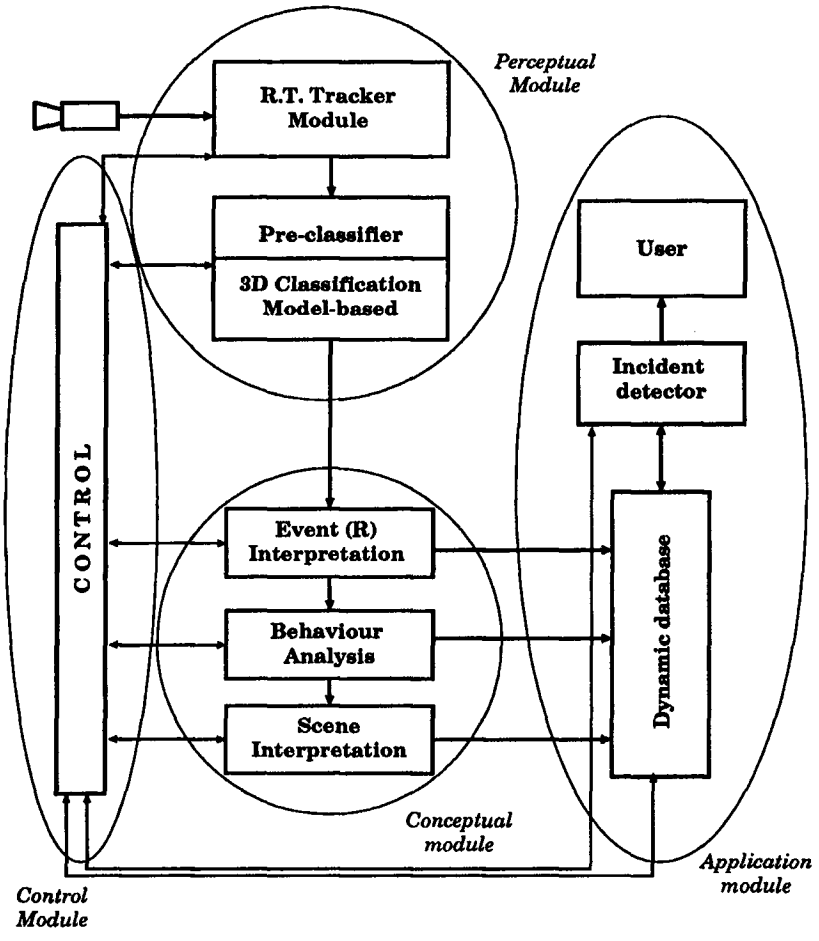


Figure 2: *Overall architecture*

3.2 Behaviour Recognition

These behaviours could be considered to be long-term compound events since behaviours are composed of a set of events related together by temporal operators. The semantic values attached to behaviours are intervals of time. Behaviours can also be constructed from other behaviours. For example, in order to recognise the GIVEWAY behaviour, i.e. an object stopping in a giveway zone, we must have recognised the STOPPED behaviour during the behaviour IN-GIVEWAY-ZONE. The former behaviour is thus built up from other behaviours while the latter two are built up from events.

Behaviours are then specified using a specially developed language which allows us to specify behaviours in terms of events and behaviours related by a set of temporal operators similar to those defined by Allen [4]. Behaviours specified in this "friendly" language are then compiled down into a relational network representation — the basic representation framework on which our reasoning is done. Our approach is then based on the propagation of temporal *intervals* through these networks. These are attached to a given object (or group as we will see later on) and represents **all** the behavioural models that are associated with an object. We will see later on that individual object networks can be interconnected.

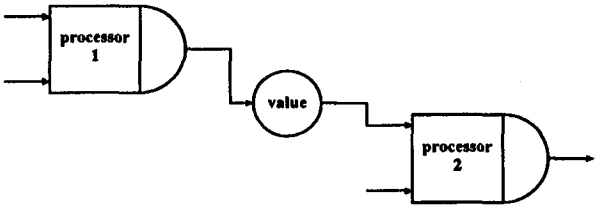


Figure 3: *Nodes of the network*

To give an idea of our approach, consider figure 3. Two sorts of nodes can be seen. There are processor nodes which correspond to the temporal operators and value nodes which are used to store values. Temporal values (intervals) are propagated along the arcs. Each processor has a defined set of terminals with particular rules, while each value node may be connected to as many other nodes as necessary. Now whenever a value transits through a connection of the nodes (i.e. when an event is recognised), it is instantaneously propagated to the other ones. Essentially these values are temporal intervals, but in some cases (i.e. for attributed events such as ENTER-REGION) pairs of values are propagated. This does not affect the fundamental mechanism.

Internal rules are defined for each type of processor node. They define :

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

It is important to note that no distinction is made between input and output nodes. Value propagation can be in any direction. This is then a means for handling incompleteness. For example, if we know that BEHAVIOUR 1 is the result of the composition of BEHAVIOUR 2 and BEHAVIOUR 3, and that BEHAVIOUR 3 and BEHAVIOUR 1 have occurred, we can deduce the occurrence of BEHAVIOUR 2 despite the absence of data concerning it.

3.3 Interactive Situations

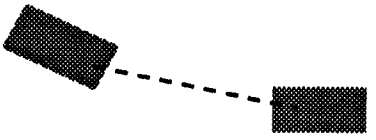
Up to now we have only considered events or behaviours associated with single objects. Though it is possible that some incidents are only concerned with a single object, the majority involve several interacting ones. As can well be imagined, the complexity increases accordingly as we could then try and relate (combine) all possible objects in the scene into groups[2]. For example, for n objects in the scene, a brute force approach would consider $\sum_{i=2}^{n} C_i^n$ different groups. For n=8, this would entail 247 possibilities. Furthermore the scenes considered are continuously evolving so we need a method to do "intelligent" grouping *dynamically*. In order to do this we now introduce our notion of *dynamic grouping* which limits the computation required, thus making the problem tractable. We have considered three types of grouping. These are :

[2]The notion of grouping originated from GEM. However it only considers *static* grouping.

- Centered around an object,
- Based on the relational events,
- Based on the spatial layout.

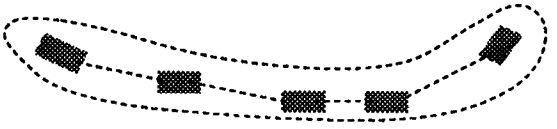
The latter two have been particularly useful in defining three types of groups necessary for our purposes. These are :

- **Binary groups**



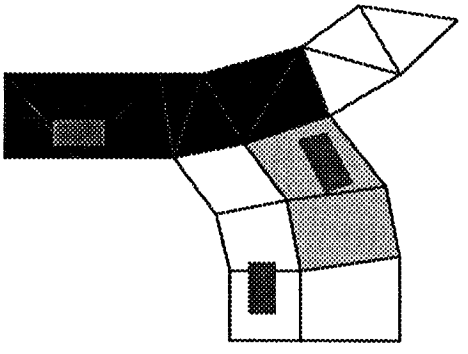
Built with the event FOLLOW. Each time an object follows another one, they are grouped together into a binary group.

- **Queue groups**



Built from the previous type of groups. When two binary groups shares an element, they form a queue group. By definition then, a queue is made up of at least three objects.

- **Giveway groups**



Some of the regions of the spatial layout are labelled as GIVE-WAY REGIONS — regions where objects should give way, while other regions are labelled as PRIORITY REGIONS in which objects are given way to. Together these two regions define a group, but this group is only created when there are objects in both regions.

Given the existence of these grouping methods, we can now define and implement group behaviour models in much the same way as for single object behaviours. Moreover these groups can generate new events. For example, the simple fact of creating a new QUEUE group can be considered as a QUEUING event.

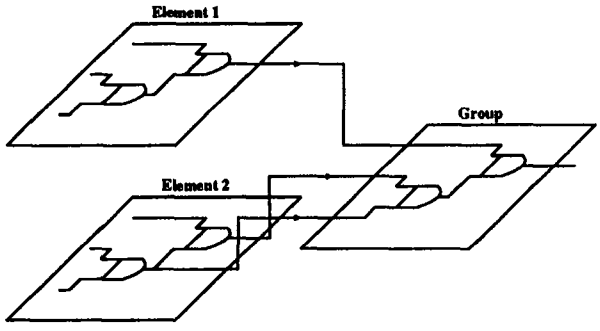


Figure 4: *Connection of networks*

Group behaviours are defined by relating the behaviours of different objects together. Practically this means connecting together the individual networks of objects in a same group (figure 4). A simple example is the BLOCKING behaviour — i.e. when another object stops in front of an object, forcing it to stop. Obviously such a behaviour cannot be attached to either object independently as we have to consider both object's behaviours and hence we connect their networks together. Propagation and hence interactive situation (multiple object behaviour) recognition is then accomplished as for behaviours.

"Intelligent" dynamic grouping and our relational network approach thus gives us an efficient way of recognising interactive situations. These situations are then filtered to recognise incidents that are of interest to the end-user.

4 Results

Event calculation and single object behaviour recognition are reasonably straightforward, however computing multiple object behaviours is a problem that can rapidly become intractable since we could potentially attempt to combine all objects together. We overcame this problem by *dynamically grouping* objects that were likely to interact with each other, thus limiting the computation and hence complexity involved.

Our brief of incident detection has been accomplished in that our system is capable of recognising multiple object behaviours that are specified as incidents by the end-user. The scenarios that we have treated contain on average fifteen objects and quite often these overlap thus resulting in partial occlusions. This resulted in some problems for the classification part of the system, with one approach being the use of Bayesian belief networks to improve the clustering and stability of the blobs.

The system currently runs about four times slower than real-time, but up to now little effort has been made on code optimisation. In fact we have associated separate Lisp processes to each object to show that our methodology is natural parallelisable, even though there is no corresponding speed up given that the workstation remains a single CPU machine.

Finally, to give some idea of how the end results are presented, a typical screen is reproduced (figure 5). This consists of a map of the scene in which we can observe the moving objects. Events are detected one or two updates (about 1/2 second) 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 described in the "incidents" window. Note that it is always possible to return to a previous incident to obtain a detailed explanation or to replay it.

5 Evaluation

Evaluation of our results can take two forms, both of which are equally important. From the end-user's point of view, what is important is that the system accomplishes a certain role that was initially specified by the end-users — they are not interested in *how* this was achieved. On the other hand, in order to provide the scientific community with some evaluation, there are several technical benchmarks against which our results can be measured. Some examples of these are the ability to handle incompleteness and uncertainty; and the ease with which the system can be used on different types of scenarios. This is what will be discussed in this section.

5.1 "End-User" Evaluation

Our brief from the "end-user" was to recognise incidents where these were defined to be

> " ...a significant change in normal behaviour ..."

This definition can still be misconstrued, but further discussion with road-traffic organisations on the scenario being treated led to the following types of incidents being detected :

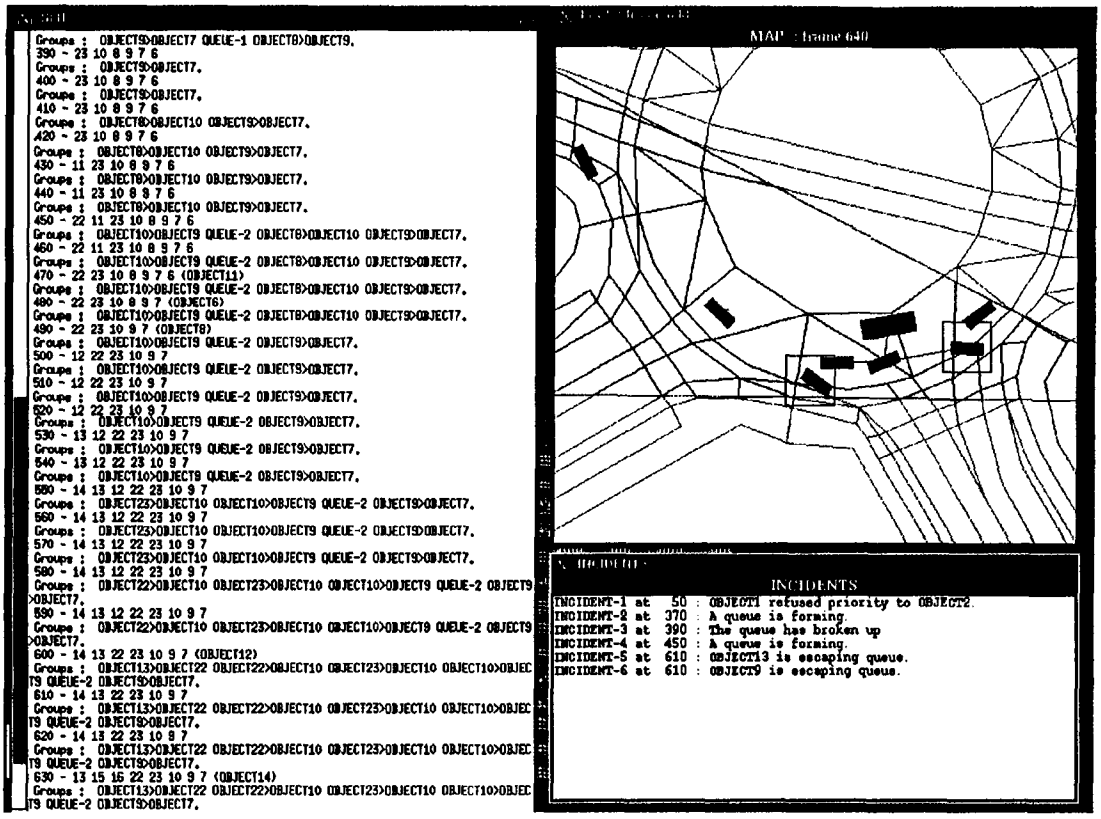


Figure 5: *Aspect of a VIEWS screen*

- Refusal of priority
- Queue forming
- Queue breaking up
- Object leaving a queue

One way to evaluate our system is to sit an end-user in front of the video, to see what incidents he recognises and then to compare his results with those achieved by our system. The results were as follows :

- Using the *perfect* simulated data, we had what could be termed a 100% + success rate in that all the end-user's incidents were recognised, and additional incidents were identified by the system. These were incidents that were not immediately obvious (though correct) since they occurred in the background of the scenario. Sometimes the results were computed too fast compared to the needs of the scene being interpreted. For example for queue detection, queue break ups were sometimes detected too quickly, as the queue reformed more or less immediately afterwards.
- Using *realistic* data, we had an 80% success rate compared to the one achieved using the simulated data. Some tuning of parameters improved the results.

What conclusions can be drawn ? Our results are very satisfactory, but are subject to tuning of the thresholds when events are computed. Obvious incidents are always detected, but more obscure or borderline ones are highly threshold dependent. Ideally some "learning" method should be incorporated. An example of this is for queue detection. In one of the scenarios we treated, we detect a queue and then a queue breaking up, more or less directly followed by a queue forming once again (with the same objects). Obviously in this case our mechanism was over sensitive.

5.2 "Technical" evaluation

Given that we had a limited number of scenarios, it was impossible to perform a true technical evaluation during the project. Nevertheless we can comment on some particular techniques :

- *Handling of uncertainty* — one of the problems in VIEWS was that of classification in the perception component (i.e. being able to correctly identify the class of an object). A result of this was that the perception component tracked each object with all possible models and left it up to the conceptual module to determine which was the most likely class. This handling of multiple class hypotheses was easily handled using the formalism we developed.

- *Handling of incompleteness* — the data used did not necessitate the need to handle incompleteness, other than to handle partial occlusions. This was done using a predictive module based on an analogical representation of space. Other forms of incompleteness could be handled given that our approach is based on relational networks, since propagation within these networks can occur in either direction.

- *Robustness of behavioural models* — it is believed that our models are robust, but these have only been tested on a limited number of scenarios of the same type.

- *Ease with which to define new behaviours* — our formalism provided us with a basic language with which new behaviours could easily be defined by relating events and/or pre-existent behaviours together with temporal operators.

- *Ease with which to move from one scenario of the same type to another* — it is believed that this should be reasonably straightforward. A point to note however is that some events/behaviours are based on the spatial decomposition and so the latter has to be taken into account also.

- *Ease with which to move from one scenario of a different type to another* — the same behaviours can be used (we intend to build up a library of behaviours), but some tuning of thresholds may be required (for example aeroplanes and road vehicles do not move at the same speed). Of course new behaviours can easily be defined. Our system was also successfully applied to one of the airport scenarios.

6 Conclusion and Future Work

Unlike the perception component where evaluation can be made in different weather conditions, different lighting conditions, etc., evaluation of the conceptual component is much more qualitative and is to a large extent dependent on how good the results of the perception component are. Nevertheless we have presented our initial evaluation of the system developed and are encouraged by the results. We should note, however, that in some cases errors (or an 80% success rate) can be unacceptable — for example if it was applied as a preventive system, then *all* potential incidents *must* be detected.

What remains to be done is a complete integration of the perceptual and conceptual modules and to make full use of the control component.

Finally we should note that there seems to be little work of a similar nature being done, the main alternative approach that we have encountered being that of Nagel [5] at Karlsruhe University. His work is however limited to single object behaviour recognition and examples given involve simpler, less complex scenarios. Other work that we are aware of considers traffic flows and thus does not reason about individual object behaviours and for this reason much of the complexity we faced does not arise.

Acknowledgements

This work was partly funded by the Commission of the European Communities under the European Strategic Program for Research and development in Information Technology (ESPRIT), project number P.2152. This project regroups Marconi Radar and Control Systems (UK), GEC-Hirst Research Centre (UK), GEC-Marconi Research Centre (UK), Atlas Elektronik (D), University of Reading (UK), Framentec-Cognitech (F), Queen Mary & Westfield College (UK), and Fraunhofer Gesellschaft (D).

References

[1] Simon King, Temporal Reasoning in *Proceedings AAAI-92 Workshop on Implementing Temporal Reasoning*, AAAI, 1992

[2] Jérôme Thoméré, Simon King, Sophie Motet and François Arlabosse. Understanding Dynamic Interactive Scenes, in *Proceedings IEEE CAIA '93*, IEEE Press, 1993

[3] Amy L. Lansky. Localized Event-Based Reasoning for Multiagent Domains, in *Computational Intelligence Journal*, 4:319–340, 1988

[4] James F. Allen. Maintaining Knowledge about Temporal Intervals, in *Communications of the ACM*, 26(11):832–843, November 1983

[5] Hans-Helmut Nagel. From Image Sequences towards Conceptual Descriptions, in *Alvey Vision Conference*, 1987