Automatic Video Interpretation: A Novel Algorithm for Temporal Scenario Recognition

Van-Thinh VU, **François BREMOND** and **Monique THONNAT**
Project ORION of I.N.R.I.A. Sophia Antipolis,
2004 route des Lucioles, BP93-06902 Sophia Antipolis Cedex, France.
{Thinh.Vu,Francois.Bremond,Monique.Thonnat}@sophia.inria.fr
http://www-sop.inria.fr/orion/orion-eng.html

Abstract

This paper presents a new scenario recognition algorithm for Video Interpretation. We represent a scenario model by specifying the characters involved in the scenario, the sub-scenarios composing the scenario and the constraints combining the sub-scenarios. Various types of constraints can be used including spatio-temporal and logical constraints. In this paper, we focus on the performance of the recognition algorithm. Our goal is to propose an efficient algorithm for processing temporal constraints and combining several actors defined within the scenario. By efficient we mean that the recognition process is linear in function of the number of sub-scenarios and in most of the cases in function of the number of characters. To validate this algorithm in term of correctness, robustness and processing time in function of scenario and scene properties (e.g. number of persons in the scene), we have tested the algorithm on several videos of a bank branch and of an office, in on-line and off-line mode and on simulated data. We conclude by comparing our algorithm with the state of the art and showing how the definition of scenario models can influence the results of the real-time scenario recognition.

1 Introduction

A problem of current focus in cognitive vision is Automatic Video Interpretation. The goal is to develop a systematic methodology for the design, implementation and integration of cognitive vision systems for recognizing scenarios involved in a scene depicted by a video sequence. An Automatic Video Interpretation System as described in Figure 1, takes as input (1) a priori knowledge containing scenario models predefined by experts and the 3D geometric and semantic information of the observed environment and (2) video streams acquired by the camera(s). The output of the system is the set of recognized scenarios at each instant. In this paper, we focus on the module of scenario recognition. The scenario recognition module takes as input the a priori knowledge of the scene and a stream of individuals tracked by a vision module.

To solve scenario recognition issues, we first propose a language to describe scenario models and second a temporal constraint resolution approach to recognize in real-time scenario occurrences. Our scenario representation is mainly based on the representation of [Vu *et al*, 2002] and inspired by the work of [Ghallab, 1996]. In this paper, we focus on the optimization of the recognition method. We first enhance the processing of temporal operators by pre-compiling scenario models to decompose them into simpler scenario models. By this way, the scenario recognition algorithm uses a linear search compared to an exponential search for non-compiled scenarios. Secondly, we propose a novel algorithm to recognize composed scenarios that takes advantages of the actors of its sub-scenarios when they are recognized instead of trying all combinations of actors as this is often the cases for similar state of the art algorithms.

We present in section 2 some related works. Our scenario representation is described in section 3. The recognition algorithm is detailed in section 4. We conclude our paper by showing experimental results to validate this new algorithm.

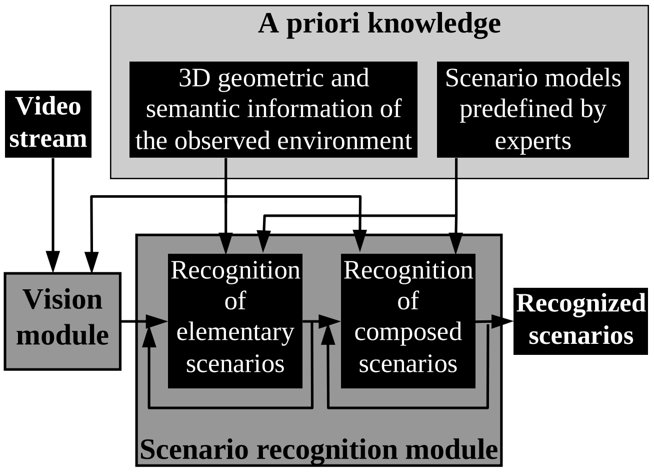
A priori knowledge
3D geometric and semantic information of the observed environment
Scenario models predefined by experts
Video stream
Vision module
Recognition of elementary scenarios
Recognition of composed scenarios
Recognized scenarios
Scenario recognition module

Figure 1. Overview of an Automatic Video Interpretation System.

2 Related works

Since the years 90s, a problem of focus in cognitive vision has been Automatic Video Interpretation. There are now several research units and companies defining new approaches to design systems that can understand human activities in dynamic scenes. Three main categories of approaches are used to recognize temporal scenarios based on (1) a *probabilistic/neural network* combining potentially recognized scenarios, (2) a *symbolic network* that Stores Totally Recognized Scenarios (STRS) and (3) a *symbolic network* that Stores Partially Recognized Scenarios (SPRS).

For the computer vision community, a natural approach consists in using a probabilistic/neural network [Oliver and Pentland, 2000]. The nodes of this network correspond usually to scenarios that are recognized at a given instant with a computed probability. For example, [Howell and Buxton, 2002] proposed an approach to recognize a scenario based on a neuronal network (time delay Radial Basis Function). [Hongeng *et al.*, 2000] proposed a scenario recognition method that uses concurrence Bayesian threads to estimate the likelihood of potential scenarios. Because video processing is very sensitive to noisy images, these probabilistic methods are useful to give an interpretation of the scene while taking into account the stochastic variations of the analysis. In our case, we use classical filtering techniques to get rid off these variations and to obtain coherent data that can be associated with symbolic values.

For the artificial intelligent community, a natural way to recognize a scenario is to use a symbolic network which nodes correspond usually to the boolean recognition of scenarios. For example, [Rota and Thonnat, 2000] used a declarative representation of scenarios defined as a set of spatio-temporal and logical constraints. They used a traditional constraint resolution technique to recognize scenarios. To reduce the processing time for the recognition step, they proposed to check the consistency of the constraint network using the AC4 algorithm. More recently, [Gerber *et al.*, 2002] defined a method to recognize a scenario based on a fuzzy temporal logic. In the same year, [Vu *et al*, 2002] present an approach to optimize the temporal constraint resolution by ordering in time the sub-scenarios of the scenario to be recognized. The common characteristic of these approaches is to store all totally recognized scenarios (recognized in the past).

Another approach consists in using a symbolic network and to store partially recognized scenarios (to be recognized in the future). For example, [Ghallab, 1996] has used the terminology *chronicle* to express a *temporal scenario*. A chronicle is represented as a set of temporal constraints on time-stamped events. The recognition algorithm keeps and updates partial recognition of scenarios using the propagation of temporal constraints based on RETE algorithm. Their applications are dedicated to the control of turbines and telephonic networks. [Chleq and Thonnat, 1996] made an adaptation of temporal constraints propagation for video surveillance. In the same period, [Pinhanez and Bobick, 1997] have used Allen's interval algebra to represent scenarios and have presented a specific algorithm to reduce its complexity.

All these techniques allow an efficient recognition of scenarios, but there are still some temporal constraints which cannot be processed. For example, most of these approaches require that the scenarios are bounded in time [Ghallab, 1996], or process temporal constraints and atemporal constraints in the same way [Rota and Thonnat, 2000].

We can distinguish two main categories of approaches to recognize a scenario based on a symbolic network: the STRS approaches recognize scenarios based on an analysis of scenarios recognized in the past [Rota and Thonnat, 2000; Vu *et al*, 2002], whereas the SPRS approaches recognize scenarios based on an analysis of scenarios that can be recognized in the future [Ghallab, 1996]. The STRS approaches recognize a scenario by searching in the set of previously recognized scenarios a set of sub-scenarios matching the scenario model to be recognized. Thus, if the system fails to recognize a scenario, it will have to retry the same process (re-verify the same constraints) at the next instant, implying a costly processing time. A second problem is that STRS algorithms have to store and maintain all occurrences of previously recognized scenarios. The SPRS approaches recognize a scenario by predicting the expected scenarios to be recognized at the next instants. Thus, the scenarios have to be bounded in time to avoid the never ending expected scenarios. A second problem is that SPRS algorithms have to store and maintain all occurrences of partially recognized scenarios, implying a costly storing space.

The method presented in this article is a STRS approach taking advantages of the SPRS approaches. The objective is to reduce the processing time (1) when searching in the past (list of previously recognized scenarios) for an occurrence of a given scenario model and (2) when trying to recognize a scenario involving several actors by avoiding checking all combinations of actors.

3 Scenario Representation

Our goal is to make explicit all the knowledge necessary for the system to be able to recognize scenarios occurring in the scene. The description of this knowledge has to be declarative and intuitive (in natural terms), so that the experts of the application domain can easily define and modify it. Thus, the recognition process uses only the knowledge represented by experts through scenario models.

Let Ω be the set of scenario models and Φ be the set of scenario instances (recognized scenarios). For a scenario model $\omega \in \Omega$ and a scenario instance $\rho \in \Phi$, we note $\omega = \omega(\rho)$ the scenario model of ρ and we note $\rho(\omega)$ the set of recognized scenarios of the model ω. A scenario is composed of four parts:

a) $\alpha(\omega)$ is the set of *actor variables* (**characters**) involved in the scenario ω. $\alpha(\rho)$ corresponds to the actors. An actor can be a person tracked as a *mobile object* by a vision module or a *static object* of the observed environment like a chair.

b) $\sigma(\omega)$ is the set of **sub scenarios** that compose ω. ω is an *elementary scenario* if $\sigma(\omega) = \varnothing$, if not, ω is called a *composed scenario*. Each sub-scenario is represented by a variable v. These variables are called *temporal variables* because their value is a recognized scenario defined on a temporal interval. We note ρ(v) a scenario instance corresponding to the value of the temporal variable v, and α(v) the set of variables corresponding to the actors of ρ(v).

c) $\varphi(\omega)$ is the set of sub-scenarios that should not occur during the recognition of the scenario ω. We call $\varphi(\omega)$ the **forbidden sub scenarios**. $\alpha_\varphi(\omega)$ is the set of *forbidden* actor variables involved in $\varphi(\omega)$ but not already defined in $\alpha(\omega)$.
$\varphi(\omega)$ and $\alpha_\varphi(\omega)$ are called the *forbidden variables*.

d) $\kappa(\omega)$ is the set of **constraints** of ω. There are three types of constraints:
- the set of **temporal** constraints, noted $\kappa^T(\omega)$, on at least one variable of $\sigma(\omega)$ and not on any forbidden variable,
- the set of **atemporal** constraints, noted $\kappa^A(\omega)$, on only $\alpha(\omega)$,
- the set of **forbidden** constraints, noted $\kappa^F(\omega)$, on any forbidden variable.

The three subsets $\kappa^T(\omega)$, $\kappa^A(\omega)$ and $\kappa^F(\omega)$ constitute a partition of $\kappa(\omega)$. We use the operator "and" to link the constraints within a set of constraints. To use the operator "or", we propose to define two similar scenario models with different constraints. An elementary scenario is only composed of a set of characters and atemporal constraints.

Figure 2 represents a model of a composed scenario "Bank attack". This scenario involves two actors, a cashier and a robber.

In our representation, any scenario ω involves at least one person, and is defined on a time interval. An interval is represented by its starting and ending times noted *start*(ω) and *end*(ω). Defining the scenarios on a time interval is important for the experts to describe scenarios in a natural way.

```
Scenario(Bank_attack,
   Characters((cashier:Person), (robber:Person))
   SubScenarios(
     (cas_at_pos,inside_zone,cashier,"Back_Counter")
     (rob_enters, changes_zone, robber,
            "Entrance_zone", "Infront_Counter")
     (cas_at_safe, inside_zone, cashier, "Safe")
     (rob_at_safe, inside_zone, robber, "Safe") )
   ForbiddenSubScenarios(
     (any_in_branch, inside_zone, any_p,"Branch"))
   Constraints(
     Temporal ((rob_enters during cas_at_pos)
               (rob_enters before cas_at_safe)
               (cas_at_pos before cas_at_safe)
               (rob_enters before rob_at_safe)
               (rob_at_safe during cas_at_safe))
     Atemporal((cashier ≠ robber))
     Forbidden((any_p ≠ cashier) (any_p ≠ robber)
               (any_in_branch during rob_at_safe))))
```

Figure 2. One example of "Bank attack" scenario is composed of four steps: (1) the cashier is at his/her position behind the counter, (2) the robber enters the bank and moves toward the front of the counter then (3) both of them arrive at the safe door and (4) nobody else in the branch during the attack. The forbidden sub-scenario (step 4) is not necessary to recognize this "Bank attack" scenario. We have included this constraint just to show the possibility of modeling forbidden sub-scenarios.

4 Scenario Recognition

The scenario recognition process has to detect which scenario is happening from a stream of observed persons tracked by a vision module at each instant. The recognition process takes also as input the a priori knowledge of the scene and the scenario models. As defined in the previous section, there are two types of scenarios: elementary and composed.

4.1 Recognition of elementary scenarios

The algorithm to recognize an elementary scenario model ω_e consists in a loop of the selection of a set of actors then the verification of the corresponding atemporal constraints $\kappa^A(\omega_e)$ until all combinations of actors have been tested. Once a set of actors satisfies all constraints $\kappa^A(\omega_e)$, we say that the elementary scenario ω_e is *recognized* and we generate an elementary scenario *instance* ρ attached with the corresponding scenario model, the set of actors and the recognition time t. The scenario instance is then stored in the list of recognized scenarios. If at the previous instant, a scenario instance ρ' of same type (same model, same actors) was recognized on a time interval $[t_0, t-1]$, the two scenario instances are merged into a scenario instance that is recognized on the time interval $[t_0, t]$.

The selection of actors leads the recognition algorithm to an exponential combination in function of the number of actor variables. However, in practice, there are few actor variables in elementary scenario models, so the recognition algorithm is still real-time.

4.2 Compilation of composed scenarios

A composed scenario is a sequence of sub-scenarios ordered in time. Each sub-scenario corresponds to a temporal variable in the corresponding scenario model. The STRS algorithms of the state of the art perform at each instant an extensive search process among all possible scenarios and sub-scenarios leading to an exponential algorithm. For example, for a given scenario $\omega = (\omega_1$ *before* ω_2 *before* $\omega_3)$; if a scenario instance $\rho_{\omega 3}$ of ω_3 has been recognized, it makes sense to try to recognize the main scenario ω. Therefore, the STRS algorithms will try all combinations of scenario instances $\rho(\omega_1)$, $\rho(\omega_2)$ with $\rho_{\omega 3}$. We propose to decompose the scenario model into a set of simple scenario models containing at most two sub-scenarios.

To compile a predefined composed scenario model ω, we define three steps: (1) build a graph with the temporal variables $\sigma(\omega)$, (2) generate intermediate scenario models for ω and (3) link the generated intermediate scenario models based on the constraints $\kappa(\omega)$.

As proposed by [Ghallab, 1996], we first build a graph which nodes correspond to the temporal variables and which arcs correspond to the temporal constraints $\kappa^T(\omega)$. The arcs are oriented and are associated with a time interval corresponding to the time delay between the ending time of the two variables. For example, the constraint c_i between v_i, v_j is associated with an interval [a, b] indicating that v_j can end in the interval [*end*(v_i)+a, *end*(v_i)+b]. The constraint *before* is associated with [1, ∞]. After building the graph (called initial graph) with all temporal constraints between temporal variables $\sigma(\omega)$, we compute the equivalent complete graph and we check the graph consistency. These two graphs are equivalent because the only difference between them is the redundancy of some constraints. Then, we simplify the complete graph by removing unnecessary arcs to obtain the least constrained graph. For any triangle ABC of the graph, an arc AC is redundant (unnecessary), if the arcs AB and BC imply a stronger constraint. The simplified graph is equivalent to the two other graphs. The initial and simplified graphs for the scenario "Bank attack" (Figure 2) are shown on Figure 3. Thanks to the simplified graph, all the temporal variables $\sigma(\omega)$ are ordered by their ending time.

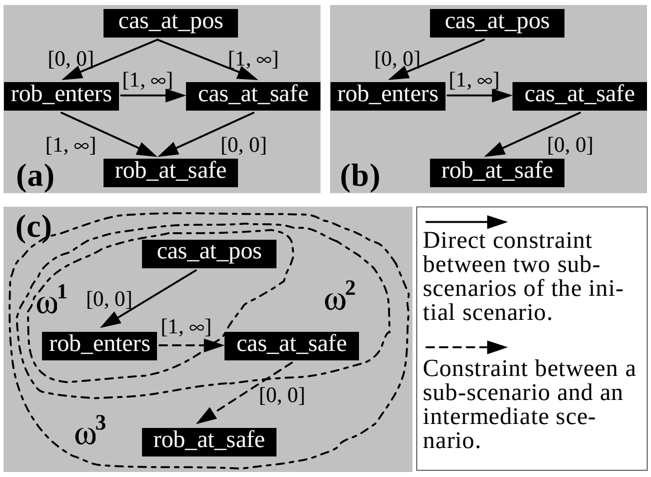

Figure 3. To compile the scenario "Bank attack", we build: (a) the graph with all temporal constraints $\kappa^T(\omega)$, (b) the simplified graph with only the necessary constraints and (c) the generated intermediate scenario models.

Second, we generate intermediate scenario models composed at most of two sub-scenarios. For each intermediate scenario model ω, we call *start* (noted $\xi(\omega)$) the first sub-scenario of ω; and we call *termination* (noted $\tau(\omega)$) the second sub-scenario of ω.

As $\sigma(\omega) = (v_1, v_2,\ldots, v_n)$ is a sequence of n (n > 2) ordered temporal variables, we generate n-1 intermediate models $\omega^1, \omega^2,\ldots, \omega^{n-1}$ as followed:

a) $\sigma(\omega^1) = (v_1, v_2)$ and
   $\sigma(\omega^i) = (v^i, v_{i+1})$ for i > 1, where v^i is a new temporal variable corresponding to the scenario model ω^{i-1},

b) $\alpha(\omega^i) = \alpha(\xi(\omega^i)) \cup \alpha(\tau(\omega^i)) = \alpha(v_1) \cup\ldots\cup \alpha(v_{j+1})$,

c) $\varphi(\omega^{n-1}) = \varphi(\omega)$ and $\varphi(\omega^i) = \varnothing$ for i < n-1,
   $\alpha_\varphi(\omega^{n-1}) = \alpha_\varphi(\omega)$ and $\alpha_\varphi(\omega^i) = \varnothing$ for i < n-1,

d) $\kappa^A(\omega^1) = \kappa^A(\omega) \cap \kappa(\alpha(\omega^1))$ and
   $\kappa^A(\omega^i) = \kappa^A(\omega) \cap \kappa(\alpha(\omega^i)) - \kappa(\alpha(\omega^{i-1}))$ for i > 1
   $\kappa^T(\omega^i)$ contains the constraints corresponding to the arcs entering v_{i+1} (i.e. $\tau(\omega^i)$) in the simplified graph.
   $\kappa^F(\omega^{n-1}) = \kappa^F(\omega)$ and $\kappa^F(\omega^i) = \varnothing$ for i < n-1.

The resulting scenario model ω^{n-1} is equivalent to the initial scenario model ω. This two scenarios have the same actor variables and equivalent set of constraints. The only difference is that the constraints of the scenario ω are verified at several intermediate levels corresponding to the intermediate scenario models as shown on Figure 4. Because any sequence of temporal variables can be ordered by their ending time, so any scenario model can be decomposed into intermediate scenarios containing only one or two temporal variables.

The recognition of compiled scenario models is described in the next section. The gain in processing time is due to the search algorithm: we just try several times to link two scenario instances instead of trying to link together a whole set of combinations of scenario instances.

```
Scenario(Bank_attack_1,   # ω1
   Characters((cashier:Person), (robber:Person))
   SubScenarios(
     (cas_at_pos, inside_zone, cashier,
                                 "Back_Counter")
     (rob_enters, changes_zone, robber,
            "Entrance_zone", "Infront_Counter"))
   Constraints((cas_at_pos during rob_enters)
             (cashier ≠ robber) ))
Scenario(Bank_attack_2,   # ω2
   Characters((cashier:Person), (robber:Person))
   SubScenarios(
     (att_1, Bank_attack_1, cashier, robber)
     (cas_at_safe, inside_zone, cashier, "Safe") )
   Constraints(
     ((start of att_1) before cas_at_safe) ))
Scenario(Bank_attack_3,   # ω3
   Characters((cashier:Person), (robber:Person))
   SubScenarios(
     (att_2, Bank_attack_2, cashier, robber)
     (rob_at_safe, inside_zone, robber, "Safe") )
   ForbiddenSubScenarios(
     (any_in_branch, inside_zone, any_p,"Branch"))
   Constraints(
     (rob_at_safe during (termination of att_2))
     (any_p ≠ cashier) (any_p ≠ robber)
     (any_in_branch during rob_at_safe) ) )
```

Figure 4. Three intermediate scenario models are generated for the compilation of the scenario model "Bank_attack", and the initial model is equivalent to "Bank_attack_3".

4.3 Recognition of composed scenarios

The recognition of a composed scenario model ω_c is triggered by a scenario template, which has been generated when the last sub-scenario ρ_t terminating ω_c has been recognized. The scenario template contains ω_c and the scenario instance ρ_t with its list of actors $\alpha(\rho_t)$ that partially instanciates $\alpha(\omega_c)$. As ω_c is composed of two sub-scenarios, only one sub-scenario ρ_s starting ω_c still needs to be found.

If such a scenario instance ρ_s has been previously recognized in the past, then we are able to finish instanciating the remaining actors of $\alpha(\omega_c)$. Thus, just a few combinations of actors need to be checked avoiding an exponential search.

The last step of the algorithm consists in verifying whether all temporal and atemporal constraints ($\kappa^T(\omega_c)$ and $\kappa^A(\omega_c)$) are satisfied with ρ_s and ρ_t. If one forbidden constraint of $\kappa^F(\omega_c)$ cannot be satisfied then the scenario ω_c is recognized and stored in the list of recognized scenarios.

4.4 Discussion

In the domain of temporal scenario recognition and among SPRS algorithms, the chronicle recognition algorithm [Ghallab, 1996] is one of the most popular. By storing partially recognized scenarios, it can speed up the whole recognition process. A partially recognized scenario corresponds to a prediction and enables to store all previous computations that do not need to be recomputed at the following instants. A main difference between the chronicle algorithm and our algorithm is that the chronicle algorithm has been developed to process scenarios defined with only one "actor" and can only recognize events detected at one time point. Thus, this algorithm is efficient for the configuration "mono-actor". However, in the configuration "multi-actors", the chronicle algorithm has to duplicate the partially recognized scenarios for each combination of actors not already instanciated. This can lead to an explosion of memory allocation and to an exponential search. Our algorithm is as efficient for the configuration "mono-actor" because we store the recognized scenarios which have been compiled. Moreover, it is efficient for the configuration "multi-actors" because the recognized scenarios do not need to be duplicated even if some actors are not instanciated. The worst case occurs with elementary scenarios because to recognize them at the current instant, all combinations of actors need to be checked. In real world applications, elementary scenarios do not contain many actor variables (less than 3) making the proposed algorithm sufficient to obtain an operational real time video interpretation system.

5 Experiments and results

To validate our recognition algorithm, we first integrated the algorithm with a vision module to obtain an operational interpretation system and then we have realized four types of tests: (1) on recorded videos taken in a bank branch and in two metro stations (one in Belgium and one in Spain) to verify if the algorithm can correctly recognize the predefined scenario models, (2) on live videos acquired on-line from cameras installed in an office, in a metro station and in a bank branch to verify if the algorithm can work robustly on a long time mode, (3) on recorded videos taken in a bank branch and on simulated data to study how the complexity of the algorithm depends on the scenario models (i.e. number of sub-scenarios and of actor variables) and (4) on simulated data to study how the complexity of the algorithm depends on the complexity of the scene (i.e. number of persons in the scene).

In the first experiment, we verify on recorded videos that the algorithm correctly recognizes several types of "Bank attack" scenarios and several types of "Vandalism against a ticket machine" scenarios in a metro station. The vandalism scenario involves in general two individuals, one looking around to check whether nobody is coming and the other one attempting several times to break the ticket machine. Table 1 shows that the predefined scenarios were correctly recognized in most of the cases. The interpretation system fails to recognize some scenarios only in the cases when the vision module misses to detect the people in the scene. We have not detected any false alarm during all the experiment. The non-detection of false alarms can be explained by the fact that the scenarios are very constrained and there are unlikely to be recognized by error.

In the second experiment, we installed the interpretation system in an office and in a bank and we connected the system to one or two on-line cameras to acquire directly live videos. In this experiment, we used the bank scenarios and we slightly modified them to use them in the office. We ran the system in the bank for few hours and continuously during 4h in the office. As in the first experiment, the scenarios were most of the time correctly recognized, showing that the recognition algorithm can work reliably and robustly in real-time and in continuous mode.

	Number of tested sequences	Average number of persons/frame	Recognition rate (%)	Number of false alarms
Bank cam. 1	10	4	80	0
Bank cam. 2	1	2	100	0
Metro cam. 2	3	2	100	0

Table 1. The recognition of temporal scenarios in videos of a bank branch and of a metro station.

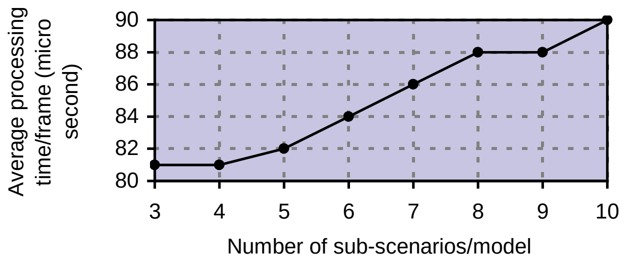

Figure 5. The processing time of the new algorithm is closely linear time in function of the number of sub-scenarios.

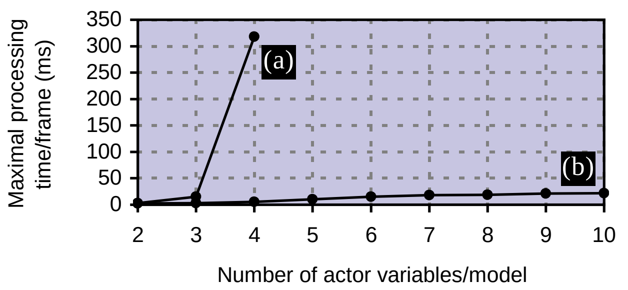

Figure 6. The processing time (a) of the old algorithm and (b) of the new algorithm depend on the number of actor variables of predefined scenario models.

In the third experiment, we studied the processing time of the recognition algorithm in function of the scenario models. First, we studied the processing time of the algorithm focusing on the resolution of temporal constraints. In this experiment (shown on Figure 5), we tested eight configurations of scenario models: the first configuration is made of scenarios containing 3 sub-scenarios and the last configuration is made of scenarios containing 10 sub-scenarios. On the bank videos containing about 300 frames, we found that the processing time of the classical STRS algorithm is exponential in function of the number of sub-scenarios, whereas the processing time of our algorithm is closely linear in function of the number of sub-scenarios.

Second, we studied the processing time of the algorithm focusing on the number of actor variables of the scenario models. In this experiment (shown on Figure 6), we tested nine configurations of scenario models: the first configuration is made of scenarios involving 2 actor variables and the last configuration is made of scenarios involving 10 actor variables. To run the algorithm with enough actors, we simulated bank videos containing 35 persons. On these videos, we found that the processing time of the classical STRS algorithm is exponential in function of the number of actor variables, whereas the processing time of our algorithm is closely linear in function of the number of actor variables.

In the fourth experiment, we studied the processing time of the recognition algorithm in function of the scene. To have a continuous variation of the scene, we simulated the scene. We built a scene environment with eight zones of interest and ten pieces of equipment. We simulated the individuals evolving in the scene at each instant. In these simulated videos, the number of individuals changed from 30 up to 240. To verify if our algorithm can recognize in real-time the predefined scenarios, we measured the maximal processing time per frame. We found that, the maximal processing time for each frame is 100ms for a scene of 240 persons. We also found that the average processing time for each frame is closely linear in function of the number of persons. Figure 7 shows several tests of this experiment to illustrate how the processing time depends on the complexity of the scene.

With the fourth experiment, we can conclude that our recognition algorithm can recognized in real-time the predefined scenarios if the number of persons/frame is less than 240. All these tests are realized on a PC linux: CPU 700MHz, 320MB RAM.

6 Conclusion

In the literature, there are two categories of symbolic approaches to recognize temporal scenarios: the STRS algorithms reasoning on the past and the SPRS algorithms reasoning on the future. First, we have shown that the STRS algorithms recognize usually a scenario by performing an exponential combination search. Then, we have explained that even if our proposed algorithm is a STRS algorithm, it checks temporal constraints nevertheless by performing a linear search thanks to a step of pre-compilation of scenarios. Second, we have also shown that the SPRS algorithms have to try all combinations of actors to be able to recognize "multi-actors" scenarios. Thanks to the pre-compilation step this drawback for our algorithm is limited to elementary scenarios. Thus, processing time can still be an issue depending on the complexity of scenarios. For these two reasons, the proposed algorithm enables the integrated video interpretation system to be real-time. Up to our knowledge, this video interpretation system is the first operational system able to recognize complex temporal scenarios.

Our future work consists in taking care of the errors and the uncertainty of the vision module. The goal is to be able to continue the interpretation of the videos even when the vision module cannot cope with the real world complexity and then to be able to recognize more complex scenarios.

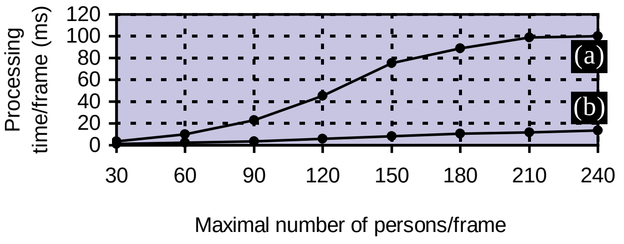

Figure 7. The (a) maximal and (b) average processing time/frame of the new algorithm depend on the number of detected persons.

References

[Chleq and Thonnat, 1996] Nicolas Chleq and Monique Thonnat. *Real-time image sequence interpretation for video-surveillance applications*. International conference on Image Processing (ICIP'96). Proceeding IEEE ICIP'96. Vol 2. pp 801-804. Switzerland. 09/1996.

[Ghallab, 1996] Malik Ghallab. *On Chronicles: Representation, On-line Recognition and Learning*. 5th International Conference on Principles of Knowledge Representation and Reasoning (KR'96), USA, 11/1996.

[Gerber *et al.*, 2002] R. Gerber, H. Nagel and H. Schreiber. *Deriving Textual Descriptions of Road Traffic Queues from Video Sequences*. The 15-th European Conference on Artificial Intelligence (ECAI'2002), Lyon, France, 21-26/07/2002.

[Hongeng *et al.*, 2000] S. Hongeng, F. Brémond and R. Nevatia. *Representation and Optimal Recognition of Human Activities*. In IEEE Proceedings of Computer Vision and Pattern Recognition, USA, 2000.

[Howell and Buxton, 2002] A.J. Howell and H. Buxton. *Active vision techniques for visually mediated interaction*. Image and Vision Computing, 2002.

[Oliver and Pentland, 2000] Nuria Oliver and Alex Pentland. *Graphical Models for Driver Behavior Recognition in a SmartCar*. Proceedings of IEEE Intl. Conference on Intelligent Vehicles 2000 Detroit. Michigan, 10/2000.

[Pinhanez and Bobick, 1997] Claudio Pinhanez and Aaron Bobick. *Human Action Detection Using PNF Propagation of Temporal Constraints*. M.T.T Media Laboratory Perceptual Section Report No. 423, 04/1997.

[Rota and Thonnat, 2000] Nathanaël Rota and Monique Thonnat. *Activity Recognition from Video Sequences using Declarative Models*. 14th European Conference on Artificial Intelligence (ECAI 2000), Berlin, W. Horn (ed.) IOS Press, Amsterdam, 20-25/08/2000.

[Vu *et al*, 2002] Van-Thinh Vu, François Bremond and Monique Thonnat. *Temporal Constraints for Video Interpretation*. The 15-th European Conference on Artificial Intelligence (ECAI'2002), Lyon, France, 21-26/07/2002.