Automatic Video Interpretation: A Recognition Algorithm for Temporal Scenarios Based on Pre-compiled Scenario Models

Van-Thinh VU, François BRÉMOND 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 with the characters involved in the scenario, with its sub-scenarios and with the constraints combining the sub-scenarios. By pre-compiling the scenario models, the recognition algorithm processes temporal constraints by decomposing complex scenarios into intermediate sub-scenarios to reduce the algorithm complexity. We have tested the recognition algorithm on several videos of a bank agency to try to recognize a scenario of "Attack". We conclude by showing experimental results of the efficiency of this algorithm for real time temporal scenario recognition.

Keyword. Automatic Video Interpretation, Scenario Recognition, Chronicle Recognition, Temporal Constraint Resolution, Scenario Representation.

1 Introduction

A problem of current focus in cognitive vision is Automatic Video Interpretation ([1], [2], [3], [4], [8], [11], [12]). 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 (AVIS) as described in Fig. 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 T. Vu, F. Bremond and M. Thonnat [12] and also based on the one of M. Ghallab and C. Dousson [4]. In this paper, we focus on the optimization of the recognition method presented in [12]. We 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 similar state of the art algorithms.

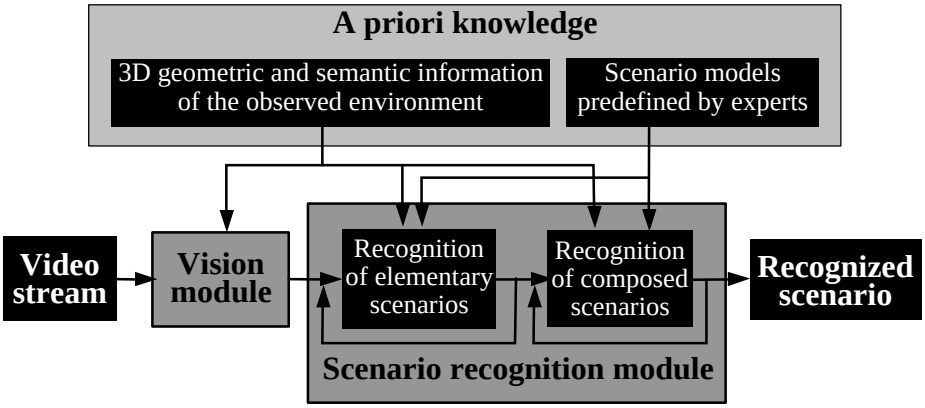

Fig. 1. Overview of an Automatic Video Interpretation System.

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.

2 Related works

For 20 years and particularly 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 of using a probabilistic/neural network. The nodes of this network correspond usually to scenarios that are recognized at a given instant with a computed probability. For example, in 1996, A. J. Howell, H. Buxton [7] proposed an approach to recognize a scenario based on a neuronal network (time delay Radial Basis Function). Two years later, F. Bremond, S. Hongeng and R. Nevatia [6] proposed a scenario recognition method that uses concurrence Bayesian threads to estimate the likelihood of potential scenarios.

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, in 2000, N. Rota and M. Thonnat [11] used a declarative representation of scenarios defined as a set of spatio-temporal and logic 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 [9]. More recently, in 2002, R. Gerber, H. Nagel and H. Schreiber [5] defined a method to recognize a scenario based on a fuzzy temporal logic. In the same year, T. Vu, F. Bremond and M. Thonnat [12] 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 of using symbolic network and to store partially recognized scenarios (to be recognized in the future). For example, in the years 90s, M. Ghallab and C. Dousson [4] have 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. Some years later, N. Chleq and M. Thonnat (1996) [3] made an adaptation of temporal constraints propagation for video surveillance. In the same period, C. Pinhanez and A. Bobick [10] 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 can not be processed. For example, most of these approaches require that the scenarios are bounded in time [4], or process temporal constraints and atemporal constraints in the same way [11].

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 ([11], [12]), whereas the SPRS approaches recognize scenarios based on an analysis of scenarios that can be recognized in the future [4]. 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) in 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 in 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 processing 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 when searching in the past (list of previously recognized scenarios) for an occurrence of a given scenario model.

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 Φ be the set of scenarios and Ω be the set of scenario models.
For each $\omega \in \Omega$ model of a scenario instance $\rho \in \Phi$, we note $\rho = r(\omega)$ and $\omega = w(\rho)$:

a) $\alpha(\rho)$ is the set of actors involved in ρ and $\alpha(\omega)$ is the set of characters (actor variables) corresponding to the actors $\alpha(\rho)$,
b) $\beta(\rho)$ is the set of sub-scenario instances that compose ρ and $\beta(\omega)$ is the set of temporal variables corresponding to sub-scenario models of $\beta(\rho)$. If $\beta(\rho) = \emptyset$, ρ is called *elementary scenario*, if not, ρ is called *composed scenario*. We note $r(v)$ a scenario instance corresponding to the value of a temporal variable v. The recognition of a scenario is a boolean value in a time interval,
c) $\gamma(\omega)$ is the set of constraints of ω expressing relations between characters $\alpha(\omega)$ and sub-scenarios $\beta(\omega)$. We note $\gamma^T(\omega)$ the set of *temporal* constraints of ω, and $\gamma^A(\omega) = \gamma(\omega) \setminus \gamma^T(\omega)$ the set of *atemporal* constraints of ω. $c \in \gamma(\omega)$ is called *temporal* constraint if c includes at least one temporal variable.

```
Scenario(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") )
   Constraints( (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) ) )
```

Fig. 2. Representation of a bank scenario "Attack": (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.

An actor can be a person tracked as a *mobile object* by the vision module or a *static object* of the observed environment like a chair. A person is represented by his/her characteristics: his/her position in the observed environment, width, velocity,…. A static object of the environment is defined by a priori knowledge (before processing) and can be either a zone of interest (a plane polygon as the entrance zone) or a piece of equipment (a 3D object such as a desk). A zone is represented by its vertices and a piece of equipment is represented by the vertices of its 3D bounding box. The zones and the equipment constitute the scene context of the observed environment [1]. Static objects and mobile objects are called *scene-objects*.

In our representation, any scenario ρ involves at least one person, and is defined on a time interval called $\delta(\rho)$. An interval is represented by its starting and ending times noted *start*(ρ) and *end*(ρ). For a temporal variable v corresponding to $w(\rho)$, we also note that *start*(v) and *end*(v) for its starting and ending times. Defining scenario on a time interval is important for the experts to describe scenarios in a natural way.

Fig. 2 represents a model of a bank scenario "Attack". This scenario involves two characters, a cashier and a robber.

This representation is similar to previous representation of scenarios [12]. The difference is that we distinguish the temporal constraints combining sub-scenarios from atemporal constraints.

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. We suppose that the persons are correctly tracked: their characteristics (their position in the scene, their height,…) are well detected and at two successive instants, two persons having the same name correspond to the same real person.

To recognize the pre-defined scenario models at each instant, we first select a set of scenario templates (called triggers) that indicate which scenarios can be recognized. These templates correspond to an elementary scenario or to a scenario that terminates with a sub-scenario recognized at the current instant. Secondly we find solutions for each of these scenario templates by looking for sub-scenario instances already recognized in the past to complete the scenario template (the resolution of a scenario template is described in section 4.1). A solution of a scenario model ω is a set of actors that are involved in the recognized scenario and the list of corresponding sub-scenario instances satisfying all the constraints of ω.

```
for each elementary scenario model ESM
   create a trigger T of type 1 for ESM
   for each solution ρe of T
      if ρe is not extensible then
         add re to the list of recognized scenarios
         add all triggers of type 2 of ρe to the list LT
      if ρe is extensible with ρ'e recognized at the previous instant then
         merge ρe with scenario ρ'e
         add all triggers of type 2 and 3 of ρ'e to LT
while (LT ≠ ∅)
   order LT by the inclusive relation of scenario models
   for each trigger T ∈ LT
      for each solution ρc of T
         add rc to the list of recognized scenarios
         add all triggers of type 2 and 3 of ρc to LT
```

Fig. 3: Overview of the scenario recognition algorithm.

We define a "trigger" as a scenario template which can be recognized. There are three types of triggers: (1) the elementary scenario models, (2) composed scenarios with specified actors and (3) composed scenarios already recognized at the previous instant. At the current instant, we initiate a list LT of triggers with all triggers of first type (*i.e.* elementary scenario models) as shown on Fig. 3. Once we have recognized an elementary scenario ρ_e, we try to extend ρ_e with a recognized scenario ρ'_e at the previous instant (the extension of a scenario is the extension of its ending time). If ρ_e can not be extended, we add the triggers of type 2 that terminate with ρ_e to the list LT. If ρ_e is extended with ρ'_e, we add the triggers of type 2 and 3 that terminate with ρ'_e. The triggers of type 2 are the templates of a composed scenario instantiated with the actors of ρ'_e and the triggers of type 3 are the templates of a composed scenario ρ_c already recognized at the previous instant and that terminates with ρ'_e. After this step, there is a loop process first to order the list LT by the inclusive relation of scenario model contained in the triggers and second to solve the triggers of LT. If a trigger contains a template of a scenario ρ'_c that can be solved (*i.e.* totally instantiated), we add the triggers of type 2 and 3 that terminate with ρ'_c. Once, a scenario is recognized, we add it to the list of already recognized scenarios indexed by a graph combining the scenario models and the list of actors to speed up the search process.

4.1 Finding solutions for a scenario model

The algorithm for finding a solution for a scenario template (trigger) consists in a loop of selecting a set of actors then of verifying the corresponding constraints until all combinations of actors have been tested. This selection of actors leads the recognition algorithm to an exponential combination in function of the number of actors. However, in practice, there are few actors in scenario models, so the recognition algorithm can still be real time.

For a scenario model ω contained in a trigger, we first check the atemporal constraints $\gamma^A(\omega)$ on actor variables. For this step, we select a set of actors corresponding to $\alpha(\omega)$. If ω is a composed scenario model and has been selected by a second type trigger (*i.e.* a partially instantiated scenario template), some actor variables can already be instantiated. Once the actors have been selected, we check all atemporal constraints. These atemporal constraints are ordered with the occurrence order of the actor variables (in a compilation phase) to speed up the recognition process [12]. If the scenario is an elementary scenario, after the verification of its atemporal constraints, the scenario is said to be recognized. If the scenario is a composed scenario, after the verification of its atemporal constraints, its actors have been instantiated but its temporal constraints still need to be verified.

To verify the temporal constraints of a composed scenario model, we extract from the set of recognized scenario instances a sub-set of recognized sub-scenarios satisfying the constraints defined in the scenario model. To search for sub-scenarios, the STRS algorithms of the state of the art (*i.e.* [12]) process usually the temporal operators by ordering the sub-scenarios in time.

Once we find a solution of a scenario model, we store the recognized scenario instance and we add to the list LT the trigger terminating with this scenario. If the scenario is an elementary scenario, we also try to extend this solution (scenario instance) with a scenario of same type (same model and same actors) recognized at the previous instant. First, if such a scenario does not exist, we just add the solution to the set of recognized scenarios. Second, if it is possible to extend the solution, we merge these two elementary scenario instances (we merge their time interval) to obtain only one elementary scenario which corresponds to a continuously recognized scenario. The extension of one elementary scenario can lead recursively to the extension of all previously recognized scenarios terminated by this scenario (as described in the previous section).

The STRS algorithms of state of the art perform at each instant a complete search process among all possible scenarios and sub-scenarios leading to an exponential algorithm. We propose to analyze temporal constraints of each scenario to order its sub-scenarios. Then the search space is reduced by decomposing the initial model into a set of simple scenarios models easy to recognize.

4.2 Decomposition of a composed scenario

A composed scenario is a sequence of sub-scenarios partially ordered in time. Each sub-scenario corresponds to a temporal variable in the corresponding scenario model. The STRS algorithms usually re-search already recognized sub-scenarios and re-verify the temporal constraints contained in any composed scenario until they find a solution for this scenario. For example, if a scenario ω is composed of three sub-scenarios: ω_1 before ω_2 before ω_3 and if ω_3 has been recognized, it make sense to try to recognize the main scenario ω. Therefore, the STRS algorithms will try all

combinations of scenario instances $r(\omega_1)$, $r(\omega_2)$ with $r(\omega_3)$ which can lead to a combinatory explosion.

If a scenario is composed of only two sub-scenarios ($\omega = [\omega_1$ before $\omega_2]$) and if the sub-scenario instance $r(\omega_2)$ has been recognized, the algorithm has to search only for one sub-scenario instance $r(\omega_1)$ in the list of recognized scenarios and this implies just a linear search. Therefore, as soon as the sub-scenario verifies a constraint, then the corresponding scenario is recognized and stored. To obtain a fast method to recognize a scenario model with a linear search algorithm, we propose to decompose any scenarios into scenarios composed at most of two sub-scenarios.

4.3 Compilation of predefined scenario models

In this section, we focus on the compilation of predefined composed scenario models. To do this, we propose an initial phase compiling a composed scenario model ω in the following steps: (1) order in time the temporal variables of ω, (2) generate intermediate scenario models for ω and (3) link the generated intermediate scenario models by using the constraints defined in ω.

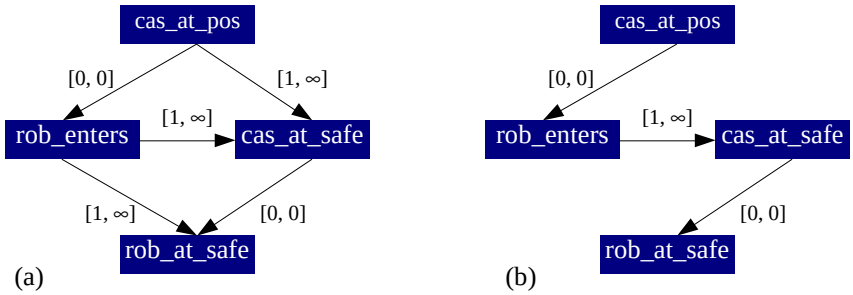

Fig. 4. Ordering in time the temporal variables of a scenario model "Attack": (a) all constraints and (b) the necessary constraints after simplifying the initial graph.

To order in time the temporal variables $\beta(\omega)$, we use a graph based method (based on [4]). The graph nodes are the temporal variables and the arcs are the temporal constraints between two variables. The arcs are oriented and are represented by time interval corresponding to the time delay between the ending times of the two variables. For example, the constraint c_i between v_i, v_j is represented by 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 represented by [1, ∞]. After building the initial graph with all temporal constraints between temporal variables of ω, we compute the equivalent complete graph (to check the graph consistency) and we simplify the graph by removing unnecessary arcs to obtain the least constrained graph. The variables are ordered by the order of the ending time. The initial and simplified graphs for the scenario "Attack" (Fig. 2) are shown on Fig. 4.

After ordering in time the temporal variables of ω, we generate intermediate scenario models composed at most of two sub-scenarios. For each intermediate scenario model ω, we call *start* (noted $\pi(\omega)$) the first sub-scenario of ω; and we call *termination* (noted $\tau(\omega)$) the second sub-scenario of ω.

Suppose that $\beta(\omega) = (v_1, v_2, \ldots, v_n)$ is a sequence of n ($n > 2$) partially ordered temporal variables. We generate n-1 intermediate models $\omega^1, \omega^2, \ldots, \omega^{n-1}$ as followed:

$\beta(\omega^1) = (v_1, v_2)$ and

$\beta(\omega^i) = (v^i, v_{i+1})$ for $i > 1$, where v^i corresponds to the scenario of model ω^{i-1},

$\alpha(\omega^i) = \alpha(\pi(\omega^i)) \cup \alpha(\tau(\omega^i))$,

$\gamma^T(\omega^i)$ is composed of the temporal constraints corresponding to the arcs entering v_{i+1} (i.e. $\tau(\omega^i)$) in the simplified graph. We can notice that several temporal operators of Allen's algebra can be ignored in this step because they are well expressed by order of temporal variables in the graph. Another task consists of modifying the constraints to adapt to the new scenario models.

$\gamma^A(\omega^i)$ is composed of the atemporal constraints involving actor variables belonging to $\alpha(\omega^1)$ for $i = 1$ and belonging to $\alpha(\omega^i)$ but not to $\alpha(\omega^{i+1})$ for $i > 1$. To avoid using the same constraint two times in two different intermediate scenarios, the atemporal constraints must involve at least one actor variable which belongs to $\alpha(\omega^i)$ but not to $\alpha(\omega^{i-1})$.

```
Scenario(Attack_1,
  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) ))

Scenario(Attack_2,
  Characters((cashier : Person), (robber : Person))
  SubScenarios((att_1, Attack_1, cashier, robber)
            (cas_at_safe, inside_zone, cashier, "Safe") )
  Constraints(((start of att_1) before cas_at_safe) ))

Scenario(Attack_3,
  Characters((cashier : Person), (robber : Person))
  SubScenarios((att_2, Attack_2, cashier, robber)
             (rob_at_safe, inside_zone, robber, "Safe") )
  Constraints((rob_at_safe during(termination of att_2))))
```

Fig. 5. Three intermediate scenario models are generated for the compilation of the scenario model "Attack", and this model is equivalent to "Attack_3".

By using this compilation method, we can obtain all composed scenario models with one or two temporal variables. In this phase, we also check the consistency of scenario models by detecting recurrent definitions (*i.e.* graph cycles) and the utilization of undefined scenario models. The recognition of compiled scenario models is identical to the recognition of not-compiled scenario models. The gain in processing time is due to the search algorithm: we just try to find one scenario instance in the list of previously recognized scenarios instead of trying all combinations of scenario instances.

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 three types of tests: (1) on recorded videos taken in a bank branch and in a metro station 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 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 to study how the complexity of the algorithm depends on the scenario models (i.e. number of sub-scenarios).

	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 using videos from a bank branch and from a metro station.

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. 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 two on-line cameras to acquire directly live videos. In this experiment, we use 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 24h 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.

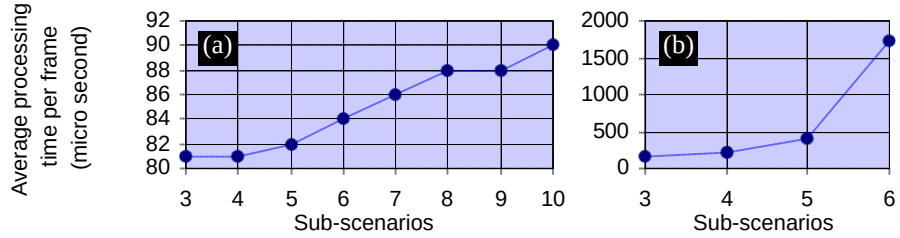

Fig. 6. The processing time (a) of the new algorithm is close to linear time and (b) the processing time of the classical STRS algorithm is exponential in function of the number of sub-scenarios.

In the third experiment, we studied the processing time of the algorithm focusing on the resolution of temporal constraints. In this experiment (shown on Fig. 6a), 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 300 frames, we found that the processing time of the classical STRS algorithm is exponential in the number of sub-scenarios (shown on Fig. 6b), whereas the processing time of our algorithm is closely linear with the number of sub-scenarios.

6 Conclusion

In this paper, we have presented a fast scenario recognition algorithm focusing on temporal constraints resolution. First, we have shown that classical STRS algorithms recognize a scenario by performing an exponential search. Second, we have described how the pre-compilation of scenarios enables the recognition algorithm to check temporal constraints by performing linear search in the list of previously recognized scenarios. Due to this new algorithm, the behavior recognition in bank monitoring becomes real time.

However, the recognition process can get into a combinatory explosion depending on the number of actors defined in the scenario models. Therefore, our current work consists of studying how scenario models can be decomposed in term of actors to limit the combinatory explosion.

Reference

[1] François Brémond. **Environnement de résolution de problèmes pour l'interprétation de séquences d'images**. *Thèse, INRIA-Université de Nice Sophia Antipolis*, 10/1997.

[2] Francois Bremond and Gerard Medioni. **Scenario Recognition in Airborne Video Imagery**. *Interpretation of Visual Motion Workshop, Computer Vision and Pattern Recognition (CVPR98)*, Santa Barbara, June 1998.

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

[4] Malik Ghallab. **On Chronicles: Representation, On-line Recognition and Learning**. *5th International Conference on Principles of Knowledge Representation and Reasoning (KR'96)*, Cambridge (USA), 5-8 Novembre 1996, pp.597-606.

[5] 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 July 2002, pp.736-740.

[6] S. Hongeng, F. Bremond and R. Nevatia**. Representation and Optimal Recognition of Human Activities**. In *IEEE Proceedings of Computer Vision and Pattern Recognition*, South Carolina, USA, 2000.

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

[8] Tony Jebara et Alex Pentland. **On Reversing Jensen's Inequality**. *In Neural Information Processing Systems 13*, NIPS 13, 12/2000.

[9] Roger Mohr et Thomas C. Henderson. **Arc and Path Consistency Revisited**. *Research Note, Artificial Intelligence*, pp225-233, vol28, 1986.

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

[11] Nathanaël Rota et Monique Thonnat. **Activity Recognition from Video Sequences using Declarative Models**. *14th European Conference on Artificial Intelligence (ECAI 2000)*, Berlin, Proceeding ECAI'00 – W. Horn (ed.) IOS Press, Amsterdam, 20-25/08/2000.

[12] 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 July 2002.