Clavis - a temporal reasoning system for classification of audiovisual sequences

Jean Carrive, Francois Pachet, Rémi Ronfard

Clavis: a temporal reasoning system for classification of audiovisual sequences

Jean Carrive & François Pachet & Rémi Ronfard

Institut National de l'Audiovisuel (I.N.A)
4, av. de l'Europe
94 366 Bry sur Marne Cedex, France
{jcarrive, rronfard}@ina.fr

SONY CSL–Paris
6, rue Amyot
75005 Paris, France
pachet@csl.sony.fr

Abstract

In the context of video indexing, we present the Clavis system in which typical video sequences of television programs are represented by templates. Templates are terminological constraint networks in which video segments coming from automatic analysis tools are represented in a description logic formalism. Templates allow to express complex classes of video sequences with temporal constraints associated to a regular expression operator. Recognizing occurrences of a template in a video program is a plan recognition problem for which efficient methods have been implemented in a constraint satisfaction problem framework. The paper describes the system and illustrates its use with several experiments that were done in the context of the DiVAN european project.

Introduction

An important step towards content-based indexing television programs is the segmentation of the video into independent meaningful units, intermediate between the shot and the complete program. This is an inherently ill-posed problem in general, since even experts fail to agree on a common vocabulary of those units, and how to compare different segmentations. The problem becomes more tractable in more constrained situations, such as a collection of videos built on a common pattern or model. In that case, we can view segmentation as a plan recognition problem, where the plans to be recognized are characteristic of a collection of videos – such as a particular broadcast news or variety show. In this communication, we introduce a representation language and a computational framework for building such abstract models of television program collections, and recognizing the models from observations.

In (Ronfard, 1997) it was first proposed to use a description logic to describe and index video shots with a rich set of film concepts. (Carrive et al., 1998) further elaborated on this idea, and extended the proposal to a general taxonomy of film events, useful in the description and the analysis of video documents in the large. In this paper, we present Clavis (Classification of Video Sequences), a system which classifies typical video sequences found in collections of television programs, using temporal compositions of film events. The classes of sequences are represented as *templates* which are terminological constraints networks. Recognizing occurrences of templates within the video is presented as a plan recognition problem and the solution proposed is an extension of the T-Rex system originally proposed by (Weida, 1995), which combines symbolic temporal relations with a regular expression operator.

The paper is structured as follows. In section 0, we describe how classes of video segments coming from automatic analysis tools are represented in a description logic formalism, and used as building blocks of the Clavis system. In Section 3, we further explain how templates are defined as terminological constraints network and how the recognition process is designed. Finally in Section 4,

we present some experiments that were done in the context of the DiVAN (Distributed Audiovisual Archives Network) project[1].

Templates: descriptions of video sequences

A lot of periodic television programs, such as newscasts, variety shows or magazines, follow a predefined scenario which presents very little variation from one edition to another. These programs are part of what is called a collection. We propose to describe typical sequences of such programs using templates and we use a plan recognition algorithm based on Weida's work on terminological constraints network (Weida, 1995).

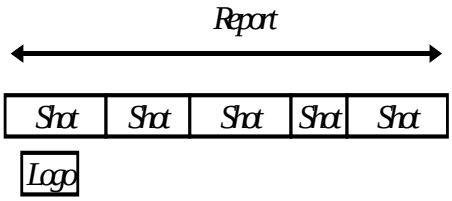
A plan recognition problem

We claim that recognising the occurrences of a template in a television program is similar to recognising which *plan* some active agent is following (Kautz, 1991) or recognising which predefined *scenario* best describes the evolution of a dynamic system (Ramaux et al., 1996), and we present this problem as a plan recognition problem. In (Fontaine, 1996), the author distinguishes four steps in a plan recognition process: opportunity, filtering, activation and discrimination. We will concentrate in this paper on the last step, which can be formulated in our case as the problem of finding in a video program made of automatically labelled segments the occurrences of a typical sequence represented by a template[2]. This recognition process often results in attributing a modality to the plan being processed, depending on whether the observations satisfy, don't satisfy or are compatible with the plan (Weida, 1995). (Ramaux and Fontaine, 1996) present a method that computes a proximity measure between the plan and the observations. We will focus in this paper on determining whether a set of observations satisfies or don't satisfies a template. It has turned out that determining whether the observations are compatible with a template is difficult and we temporarily put this task to one's side.

Following (Weida, 1995), we define a template as a terminological constraint network. The vertices of the network are associated with *concepts* which describe classes of video segments coming from audio or video analysis tools. The edges of the network are temporal constraints which have to be respected by the *observations*, *i.e.* the video segments coming from the analysis tools. In addition to Weida's formalism, we define an iteration operator "*" which expresses a contiguous sequence of video segments. For instance, a simple template may describe a report as a sequence of consecutive shots with a logo appearing during the first shot of the sequence. This template is defined by the following expression:

Report

nodes:	s1 [*Shot*]
	s2 [*Shot**]
	s3 [*Logo*]
constraints :	s1 {*meets*} s2
	s3 {*starts* $\vee$ *during* $\vee$ *finishes*} s1
	s1 {*starts*} this
	s3 {*finishes*} this



Classification of basic film events

Everything that appears on the screen, or is heard in the soundtrack constitutes an *event*. In this paper, we will consider that the temporal part of an event is only a time interval – for example a start point and a duration – identifying the temporal occurrence of the event within the video. Much like in grammars, we assume that a video is composed of several terminals, which we call segments. These

---

[1] DiVAN is the Esprit project N° 24956.

[2] We choose the term "template" because *scenario* or *plan* ("plan" means "shot" in French) are ambiguous in the television domain.

events are terminal because they are not themselves composed of other segments. An event can be a particular shot, which is what is filmed in one shot of the camera, a segment of music, a gradual transition between two shots as a dissolve, etc.

When applied in the context of DiVAN, the complete indexing process consists of the following steps: 1) initial segmentation of the audio and video track, producing at least two separate segmentation layers (usually more); 2) feature extraction and classification of segments, based on learned statistical models; 3) symbolic classification, using DL descriptions; and 4) recognition of composite events. In this paper, we assume the results from 1) and 2) provide a set of terminal events and focus on step 3) and 4).

Description Logics Description logics (Nebel, 1990) form a family of knowledge representation languages which derive from works on semantic networks and frame languages. In a description logic, *concepts* represent sets of *individuals*, and *roles* represent binary relations between individuals. Concepts can be seen as unary logical predicates, as roles are similar to binary logical predicates. Concepts can be described by syntactic operators as intersection (AND), union (OR), restrictions on the domain of a role or on the cardinality of a role. Concepts (and sometimes roles) are organized into a taxonomy according to a generality link – a *subsumption* link. Computing the subsumption relation between two concepts is one of the principal task of a description logic system. *Instantiation* is one other important operation, which compute the set of concept an individual belongs to. *Primitive* concepts are defined with necessary – and not sufficient – conditions, as *defined* concepts are defined with necessary and sufficient conditions.

Taxonomy of film events Using the CLASSIC system (Borgida, 1989), we define classes of events as concepts, focusing of classes of events which can be automatically recognized by audio and/or video analysis tools, such as those which are integrated in the DiVAN prototype: segmentation into shots, face regions and text regions detection, music/speech discrimination, jingle detection in the audio track, for example.

Figure 1 shows five concepts corresponding to cases where a face region can be detected clearly. These concepts are derived from terms of a cinematographic vocabulary, called “shot values”, and range from close-up (CU), where the face occupies approximately half of the screen, to the long shot (LS), where the human figure is seen entirely, and the face occupies around ten percent of the screen. Intermediate shot values are the medium shot (MS), the medium-close-up (MCU) and the medium-long-shot (MLS) (Thompson, 1998). Shot values are usually defined in relative and imprecise terms, based on the distance of the subject to the camera. We use the fact that the apparent size of faces on screen vary inversely with their distance to the camera to provide a computable definition of shot values. The ratio of the width of the face to the width of the frame is used to classify a shot among the five concepts. For example, the CLASSIC definition of a MCU shot is the following:

```
(cl-define-concept 'MCU-Face
    `(and
        face
        (all face-ratio (and (min ,(/ 1 6)) (max ,(/ 1 2))))))
```

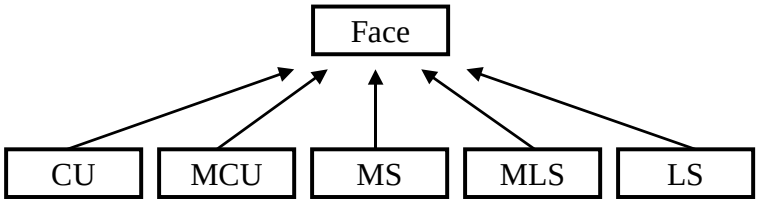


Figure 1: shot values

The algorithm used in DiVAN to segment a video into shots also detects two different classes of progressive transitions between shots, namely dissolves and wipes. In a dissolve, the existing image is

progressively replaced by superimposing a new image. In a wipe, a geometric pattern – often a simple line – erases the existing image and reveals the new one. This editing effects are illustrated by Figure 2.

Figure 2: progressive transitions between shots

In may happen in some collections of documents that low-level features can directly provide high-level information. For example, in the five editions of the "France 3" evening newscast "Soir 3" from the DiVAN corpus, when some text is displayed at the bottom left of the screen during a medium close-up shot of a character, the text always refers to the character on-screen, mentioning its name and sometimes its function or role (see Figure 3). Thus, a "named person shot" can be defined by the very simple CLASSIC expression:

```
(cl-define-concept 'NamedPersonShot
    '(and
        MCU
        BottomLeftTextShot)))
```

Figure 3: example of a *NamedPersonShot*

When a shot is classified both as a MCUShot and as a BottomLeftTextShot, it is automatically classified as a NamedPersonShot. The definition of this concept is specific to the "Soir 3" newscast. Specific concepts are defined for each collection, starting from generally defined concepts as MCU. In other words, a general taxonomy of concepts is specialized for each collection of programs.

Constraint networks

A template is a temporal constraint network whose vertices are associated with concepts defined in a description logic formalism, or with other templates. A vertex associated with a concept is called an "elementary" vertex and a vertex associated with a template is told a "composed" vertex. An iteration operator "*" is defined which can be applied on the vertices of the network which are associated with concepts or with certain types on templates. This types of templates will be discussed later. The "*" operator can be compared to the "+" operator in regular expressions, as it indicates a contiguous sequence of at least one element. Non iterated vertices are told "simple" vertices. This iteration operator indicates a sequence of contiguous events. This leads to four different types of vertices:

- simple elementary vertices (v_C)
- simple composed vertices (v_T)
- iterated elementary vertices (v_{C*})
- iterated composed vertices (v_{T*})

The edges of the network are temporal constraints. In the current implementation of the system, these temporal constraints are temporal relations in the full interval algebra. A template is recognized – or satisfied – if and only if :

- each simple elementary vertex v_C is matched with an observation which is an instance *C*;
- each simple composed vertex v_T is matched with a subset of the observations which satisfies *T*;
- each iterated elementary vertex v_{C*} is matched with a subset of the observations forming a contiguous sequence of instances of *C*;
- each iterated composed vertex v_{T*} is matched with a subset of the observations forming a contiguous sequence of satisfied templates *T*;
- the matching respect the temporal constraints defined in the template.

When a set of observations satisfies a template *T*, it is said to be an instance of *T*. Without iterated vertices, the problem of template recognition comes down to find a matching between the constraint network and the observation network. This network matching problem has been proved to be NP-hard (Weida, 1995). Recognizing templates with iterated vertices lead to matching vertices of the template with sub-networks of the observation network. In order to avoid a combinatorial explosion, we designed methods which are very efficient for the type of cases we have to deal with. An overview of these methods is presented in section 0.

Temporal constraints

Several representations of time may underlie a temporal constraint network. The two main categories are the time point algebra (Vilain et al., 1989) and the time interval algebra (Allen, 1983). Complete constraint propagation in the latter representation is NP-hard and tractable subclasses of this algebra have been proposed (Nebel et al., 1994; Drakengren et al., 1997). Following (Weida, 1995) we have chosen the full interval algebra with a 3-path consistency constraint propagation algorithm which is potentially non complete. The reason why we choose this formalism is that when we started this work we didn't have a precise idea on what would be the most appropriate representation of time. We thus adopted this very general and expressive formalism. In a second step, we should work on determining what are the temporal constraints we really need. For example, it appeared on the one hand that numerical constraints such as "a segment of music which starts less than 30 seconds before a shot", as in (Aigrain, 1997), would enhance the powerfulness of our system. On the other hand, the full expressiveness of Allen's algebra didn't appear yet to be necessary.

The temporal extension of an instance *t* of a template *T* can be specified in the template definition by setting temporal constraints between *t* – or more precisely what will be *t* when *T* will be recognized – and its components. The instance of a template is designed by "this" in the template definition. For example, the template illustrated by Figure 4 defines a "musical shot" as a shot which appears during a musical segment. The temporal extension of a musical shot is naturally set as being equal to the observed shot. Note that any disjunction of Allen's relations may be set between the instance of a template and its components.

MusicalShot

nodes: s1[*Shot*]
s2[*Music*]

constraints: s2 *during* s1
s1 *equals* this

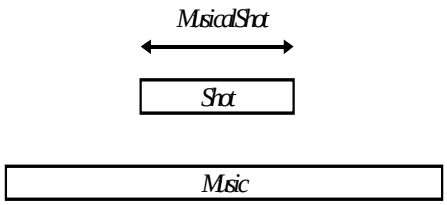


Figure 4: temporal extension of an instance of template

Iterated sequences

An iterated vertex v_{C*} or v_{T*} in a template represent a contiguous sequence of observations which are instances of *C* or instances of *T*. An iterated sequence contains at least one element, and the elements of an iterated sequence are contiguous, which means that two successive elements must be related by the Allen's *meets* relation. The temporal extension of an iterated sequence is defined as the temporal union of the temporal extensions of its elements. This implies that the temporal extensions of those elements are known, which lead to limit the types of templates which can be associated to an iterated vertex (see section 0). These types of templates are what we call "bounded" templates, *i.e.* templates whose temporal extension of instances may be computed. Roughly, a template is bounded if some of its vertices are in such a temporal relation with "this" that the temporal extension of its instances may be computed from the temporal extension of its components. This vertices have to be associated with event concepts or with other bounded templates. The temporal relations that allow to compute the starting point are *starts*, *is-started*, *meets*, *equals*, and the temporal relations that allow to compute the ending point are *finishes*, *is-finished*, *is-met*, *equals.*

The Figure 5 illustrates an iterated sequence of the template *MusicalShot* illustrated by Figure 4. Note is this example that the same segment of music was used to recognize each of the shots as instances of the *MusicalShot* template.

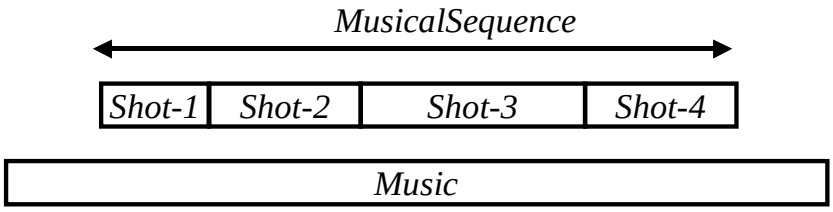


Figure 5: example of an iterated sequence

Other constraints

Once the general architecture of a template is designed as a temporal constraint network whose vertices are associated to concepts in a taxonomy of audiovisual events or with other templates, and whose vertices may be iterated in some cases, it is possible to define other constraints on its vertices. For example, we define a "no … between" constraint telling that for a template to be recognized, there should be no instance of some concept *C* or some template *T* between the observations matched with two of the vertices of the template. For example, the template illustrated by Figure 6 defines a report as the sequence of shots that is shown between two consecutive "jingle" shots, a jingle shot being a shot during which some instance of *Jingle* is heard.

Report

nodes: s1[*Shot*]
s2[Shot*]
s3[Shot]
s4[Jingle]
s5[Jingle]

constraints s1 *meets* s2
s2 *meets* s3
s4 *during* s1
s5 *during* s3
s2 *equals* this
no *Jingle* between s4 s5

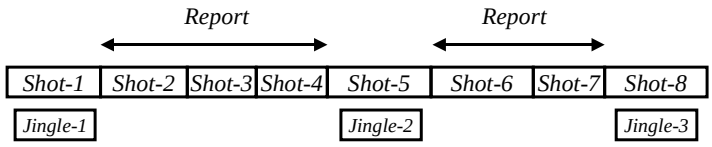


Figure 6: example of template with a "no … between" constraint

Other constraints could be easily defined, which would be taken into account by the constraint solver during the recognition process. It might be for cardinality constraints on the number of elements of an iterated sequence, or even numerical constraints on the temporal dimension of vertices, as "two instances of *Jingle* separated by at least 30 seconds". Note however that these constraints would not be taken into account during the construction of the temporal constraint network, and thus that inconsistencies would not be detected. In this case, the recognition process would scan the whole search space before answering that there are no solutions.

CSP techniques for template recognition

The problem of recognizing instances of a template *T* among the observations is expressed as a constraint satisfaction problem (CSP). Roughly, each vertex of *T* gives a *variable* of the CSP. Each variable takes its values from its *domain*, a finite set of possible values. The solutions of the problem are expressed as a set of *constraints* which are boolean functions whose arguments are variables.

In this section, we give an overview of the implementation of the Clavis system, focusing on the recognition of iterated sequences. The implementation deeply relies on BackJava (Roy et al., 1999), a (CSP) framework which allows to implement specific classes or variables, constraints and even domains or heuristics as subclasses of general purpose predefined classes. Thus, general mechanisms as arc-consistency can be used, as specialized filtering methods can be defined. In the system presented here, we use general unary constraints for checking that a simple elementary vertices are matched with observations that are instances of the concept associated with the vertex, general binary constraints for temporal constraints between simple vertices and specialized filtering method for temporal constraints which implies iterated vertices. These methods are sketched below. Finally, we let the default resolution mechanism of BackJava realize the recognition process, *i.e.* we let it choose when it should instantiate a new variable, which variable to choose, when it should backtrack, etc. The time responses we get during the experimentations we did – a few seconds for templates with iterated composed vertices with about 200 observed events – were quite encouraging and we didn't try to optimize the resolution phase.

The most complicated part of the implementation concerns iterated vertices of templates. We describe what is done with iterated elementary vertices. Iterated composed vertices associated with a template *T* are managed in a similar way after all instances of *T* have been recognized. Each iterated elementary vertex v_{C*} gives an "iterated variable" of the CSP. The first problem is to represent the domain of the variable, *i.e.* the set of iterated sequences of instances of *C*. The number of such sequences can be very important. For example, for an observation network made of *N* consecutive shots, there are $\frac{N(N+1)}{2}$ distinct iterated sequences of shots. In this case, illustrated by Figure 7, all sequences of shots may be represented as sub-sequences of the biggest sequence of shots, which is called a "maximal" sequence. A maximal sequence of instances of *C* is an iterated sequence of instances of *C* where no observation is both an instance of *C* and is in the *meets* relation (respectively the *is-met*) relation with its first (respectively its last) element. Each sub-sequence is uniquely determined by its size and the index of

its first element in the maximal sequence. An indexing function is used, which associates a unique integer between 0 and $\frac{N(N+1)}{2}$ -1 to all sub-sequences of a maximal sequence of size *N* according to their size and the index of their first element in the maximal sequence. The domain of an iterated variable is thus represented as a list of integers which is internally implemented as a list of intervals. This kind of integer variables are already implemented in BackJava which takes in charge basic operations on intervals, like union or intersection.

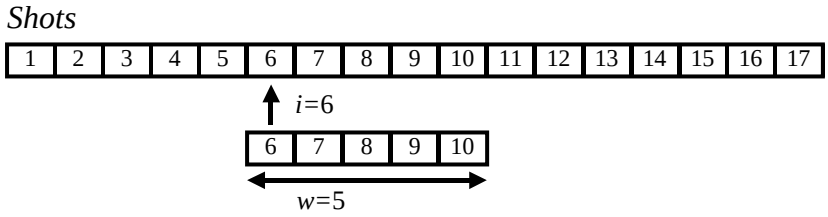


Figure 7: representation of sub-sequences

In order to compute the domain of an iterated variable representing an iterated vertex v_{C*}, one's must first compute the set of all maximal sequences of instances of *C*. This computation could be very expensive in the general case, but is quite acceptable for the kind of cases we have to process.

Temporal constraints which imply an iterated variable are implemented using a set of several filtering methods. Each of this method is intended to reduce the domain of the iterated variable in a given context. The general principal is the following. Let $C_R(v, v^*)$ be a temporal constraint which imposes hat the values of the two variables v and v* are in the *R* temporal relation, v* being an iterated variable, v being either iterated or not. When v is instantiated, *i.e.* when the solver chooses a value for v, some filtering methods are called which suppress from the domain of v* the iterated sequences which don't respect the *R* relation.

Let us take as example a template which specifies that a jingle must precisely *meets* an iterated sequence of shots. The *meets* relation is expected to be frequently encountered in templates, and thus a specialized filtering method has been designed for it. The v_J variable represents the jingle and the v_{S*} variable represents the shot sequence. The observations are illustrated by Figure 8. During the recognition process, the solver may choose to affect the *jingle-1* to v_J. In this case, the *doMeets* filtering method is called, which suppresses from the domain of v_{s*}. all the sequences which don't start with the sixth shot, as illustrated in the figure. Similar methods are designed for other cases which are expected to frequently happen, as the *starts*, *equals* or *finishes* temporal relations.

Another set of 13 methods are designed to cope with any temporal constraint, one for each of the Allen's basic temporal relation. For the *r* Allen's basic relation, the filtering method *doNot-r* is implemented, which takes as argument an iterated variables v* and an observed event *e*, and which suppresses from the domain of v* all the sequences which are in the *r* relation with *e*. In order to process a $C_R(v, v^*)$ constraint in the case where *R* is any disjunction of Allen's basic relations, the *doNot-r* method is called for all the Allen's basic relation *r* which are not part of *R*.

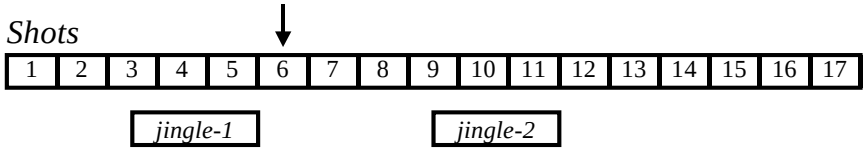


Figure 8: filtering iterated variables

Template subsumption

It would be appreciable if the template library was hierarchically organized according to subsumption links, as are concepts in a taxonomy. In (Weida, 1995), recognizing instances of templates among the

observations amounts to testing subsumption between templates, as plans and observations are roughly the same kind of temporal constraint networks. Unfortunately, the iteration operator we introduce in the template definition language results in an important complexity in the computation of subsumption between templates, as the '*' may appear in both the subsuming and the subsumee template. This implies that computing subsumption would necessitate to find a matching from sub-networks of the subsuming constraint network to sub-networks of the subsumee constraint network. Consider for example the templates T_1 and T_2 illustrated by Figure 9. T_1 subsumes T_2, as any set of observations recognized as an instance of T_2 would also be recognized as an instance of T_1.

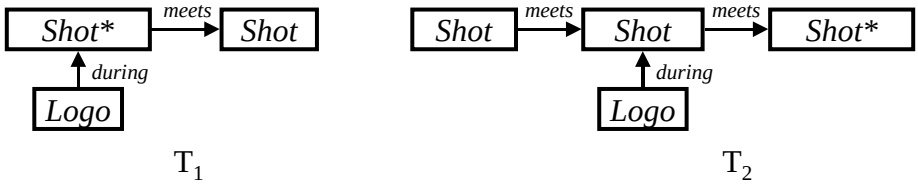


Figure 9: templates subsumption

Experimentation

We present in this section some experimentation we made on different broadcast news. Some of the documents come from an annotated corpus which was provided by INA and the AIM[3] group, the other are part of the corpus of the DiVAN project. The results presented here rely on reference segmentations established for evaluation of analysis and classification tools of the DiVAN project (Bouthemy et al., 1999). The newscasts presented here fall into two main categories:

- Traditional newscasts, alternating between the anchor person in the studio, and pre-recorded stories;
- Short newscasts composed of a small set of pre-recorded stories separated by jingles and/or graphics.

In order to recognize reports from this two types of newscast, two different methods are used which use different templates. It should be noted that similar classes of events – similar concepts – in two different newscast may have different definitions at the signal level. For example, the name of a character being filmed appears differently in a "Soir 3" and in a "France 2" evening newscasts, as well as the place of the logo is different (see Figure 3 above and Figure 10 below).

Reports of "France 2" newscast

In an edition of the "France 2" evening newscast, report sequences alternate with shots showing the anchorman in a studio. The anchorman can be filmed from different cameras. The logo of the channel always appear during report sequences on the bottom right of the screen, and never appears on studio shots, except once[4]. During report sequences, a text inscription on the bottom left of the screen always indicates the name of the person being filmed as a text inscription on the bottom right of a medium close-up shot always indicates the name of the location where the action takes place. Detecting these types of shots is interesting for at least two reasons. First, it may help to temporally structure the report sequence itself. Second, these shots may be used for summarize the report sequence, by for instance preferably select keyframes coming from these shots.

By exploiting the results of three analysis tools, namely a logo, a text region and a face region detection tool, four classes of shots can be defined for this newscast. These four classes are illustrated by Figure 10 and are organized according to subsumption links as shown in the figure.

---

[3] Action Indexation Multimedia
[4] We do not take this shot into account here

Figure 10: the four classes of shots from "France 2" evening newscast

We present here an edition of a "France 2" newscast coming from the AIM corpus. This document contains 157 shots, including 15 studio shots, 7 shots of named persons and 11 shots of named places. The temporal order of the shots is shown Figure 11. In the figure, the missing shots are report shots which are not shots of named place or named person.

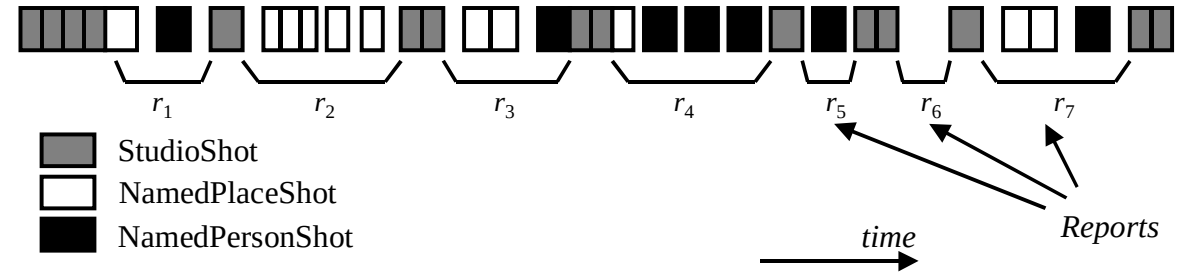


Figure 11: temporal order of shots in "France 2" example

In this example, we define a report as being what is shown between two shots of the anchorman. We thus extract the reports from the list of shots by defining the following template:

Report

nodes:	s1[StudioShot]
	s2[*ReportShot**]
	s3[*StudioShot*]
constraints:	s1 *meets* s2
	s2 *meets* s3
	s2 *equals* this

This template specifies that a report is recognized when it occurs that a studio shot is followed by a sequence of report shots which is followed by another studio shot, and that the temporal extension of

the report is equals to the temporal extension of the report shot sequence, *i.e.* it excludes the studio shots. This template is illustrated by Figure 12.

Figure 12: example of a "France 2" report

The recognition of this template gives 7 reports, as expected. The recognized reports are shown Figure 11. We then try to establish which reports contain at least one shot of a named person. The following template is used for this purpose:

ReportWithNamedPerson

nodes:	s1[*Report*]
	s2[*NamedPersonShot*]
constraints:	s1 {*starts finishes during equals*} s2
	s1 *equals* this

This template specifies that the reports we are looking for are reports "during" which a shot of a named person occurs, the meaning of "during" being given by the disjunction of Allen relations: {*starts* ∨ *finishes* ∨ *during* ∨ *equals*}. This relation states that the shot of the named person can be at any place in the report and can even temporally equals the report, which would be the case if the report is composed of a single shot of a named person – intervention of a foreign correspondent, for example. The temporal extension of such a report is obviously the same as the original report, as stated by the last constraint of the template definition.

The reports r_1, r_3, r_4, r_5 and r_7 of Figure 11 are recognized as reports with a named person. Note that the r_4 report contains three distinct shots of a named person. Thus, there are three different ways to recognize r_4 as a report with a named person. The recognition process gives three responses for r_4 t o such a report, and a post-treatment is needed in order to keep only one answer.

Reports recognized from audio jingle occurrences

Several short newscasts share a common very simple temporal structure: reports are only separated by jingles. This is the case for example for the French channels M6 (the "6 minutes" newscast), Canal+ and Arte (the "8 ½" newscast). Most of time, the jingles of such newscasts are sequences of very similar images accompanied with very similar sound samples. We are interested here in the case where two analysis algorithms are applied, one providing a segmentation of the visual part into shots and the other detecting occurrences of jingles within the audio track. Jingles and shots are temporally independents, as shown in Figure 13.

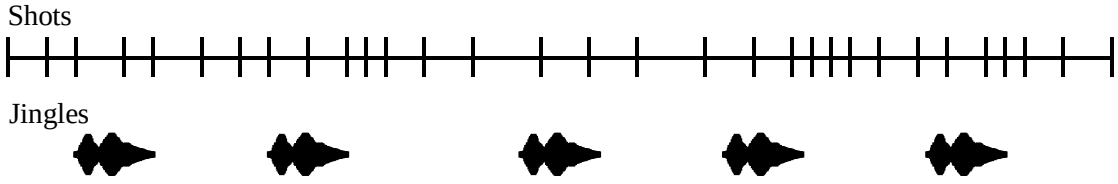


Figure 13: shot segmentation and jingle detection

We define in this case a report as a sequence of shots which is delimitated – in a way that will be précised later – by two jingles. We consider the first jingle as being part of the report in order to be

able to build sequences of reports, *i.e.* to recognize the *Report** part of the whole newscast template. We choose to include the first jingle as a jingle is often announcing the report and may contain visual or audio information on the report. A report is illustrated Figure 14. The first and the last reports constitute special cases, which are dealt with separately.

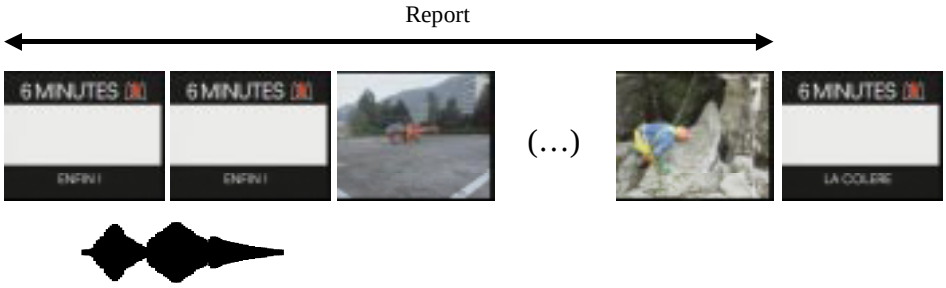


Figure 14: example of a "M6" report

We first define the whole temporal structure of the newscast, which is simpler that the definition of the report, and which is given by the following template:

Newscast

nodes:	s1[*FirstReport*]
	s2[*Report**]
	s3[*LastReport*]
constraints:	s1 *meets* s2
	s2 *meets* s3
	s1 *starts* this
	s3 *finishes* this

This template indicates that the newscast is composed of its first report followed by a sequence of reports followed by its last report, and that its temporal extension is the temporal union of all the reports. The template which defines a report is a bit more complicated, in order to handle all the possible relative temporal positions of the shots which are part of the jingles with respect to the audio part of those jingles. The definition of this template is the following:

Report

nodes:	s1[*Shot*]
	s2[*Shot**]
	s3[*Shot*]
	s4[*Jingle*]
	s5[*Jingle*]
constraints:	s1 *meets* s2
	s2 *meets* s3
	s1 {*overlaps starts is-started is-during*} s4
	s3 {*overlaps starts is-started is-during*} s5
	s1 *starts* this
	s2 *finishes* this
	no *Jingle* between s4 s5

At least three shots are necessary to recognize a report. Shots labelled s_1 and s_3 in the template delimitate the report. The temporal relation {*overlaps* $\vee$ *starts* $\vee$ *is-started* $\vee$ *is-during*} which constraint s_1 and s_3 stands for "the earliest shot which temporally intersect the jingle". Roughly, a report is said to last from the beginning of one jingle to the beginning of the next jingle.

The way the temporal constraints are set in the template ensures that recognized reports form a contiguous sequence. The last constraint imposes that there is no jingle between the jingles matched with s_3 and s_4, *i.e.* s_3 and s_4 are matched with consecutive jingles. Figure 15 shows an example of such a report. The matching of the template nodes are indicated on the figure, as are indicated the observed constraints between the shots which bound the report and the jingles.

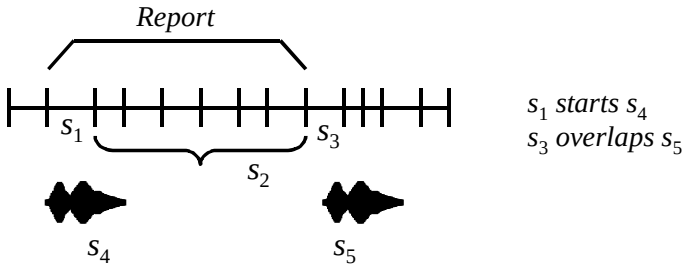


Figure 15: example of a report recognized from jingles

First and last reports are defined in a similar way, assuming that the first and the last shots have been labelled as *FirstShot* and *LastShot*. The definition of the template of the first report is given bellow:

FirstReport

nodes:	s1[*FirstShot*]
	s2[*Shot**]
	s3[*Shot*]
	s4[*Jingle*]
constraints:	s1 *meets* s2
	s2 *meets* s3
	s3 {*overlaps starts is-started is-during*} s4
	s1 *starts* this
	s2 *finishes* this
	no *Jingle* between s1 s3

At a first glance, it may not appear necessary to explicitly refer to the shots which bound the report – shots indicated by s_1 and s_3 in the report template – and a report may seem to be more simply defined as a sequence of shots "between" two consecutive jingles, which would naturally lead to define the template:

NaiveReport

nodes:	s1[Shot*]
	s2[Jingle]
	s3[Jingle]
constraints:	s1 {*is-started is-overlapped*} s2
	s1 {*meets overlaps*} s3
	s1 *equals* this
	no *Jingle* between s2 s3

Now consider the shots and the jingles illustrated by Figure 16. In the figure, shots which can be the first shot of a naïve report are indicated by a star and shots which can be the last shot of a naïve report are indicated by a diamond. Thus, there are 9 possible naïve reports, as there is only one report when applying the previous template. Moreover, this template doesn't take into account some special cases, as the case where the audio jingle is totally temporally contained in one single shot.

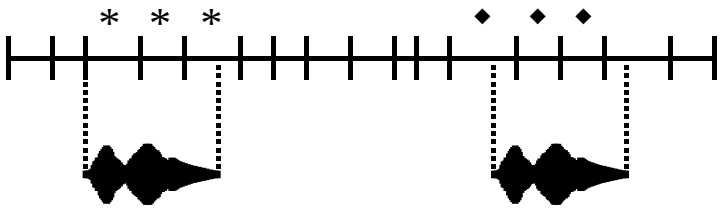


Figure 16: multi occurrences of a naïve report

We applied the newscast template to a "M6" newscast[5] containing 174 shots and 10 jingles. The recognition process gives exactly 1 matching for the newscast, which is made of 1 first report, a sequence of 9 reports, and 1 last report, which means that all reports have been recognized and that each report was recognized is only one way. On the other side, the recognition of the "naïve" report template gives 485 reports.

Other types of programs follow similar structures. For instance, programs of the "Top A" variety show from the DiVAN's corpus present successive songs which can be segmented by only using the applause occurring at the end of each song. Delimitating the temporal boundaries of a song is however more difficult than those of a "M6" report, because applause can temporally overlap the music.

Reports recognized from progressive transitions

We compare in this section two different ways of recognizing reports from five editions of "Soir 3", the evening newscast of the french France 3 channel. Four of these documents are part of the DiVAN corpus and the last one comes from the AIM corpus. A report of this newscast is defined as the sequence of shots that is showed between two consecutive shots of the anchorman. This definition is taken as the reference. The first method for recognising reports uses only one concept, *AnchorManShot*, and one template:

Report	
nodes:	s1[*AnchorManShot*]
	s2[*AnchorManShot*]
constraints:	s1 *before* s2
	s1 *meets* this
	s2 *is-met* this
	no *AnchorManShot* between s1 s2

Most of time, reports in a "Soir 3" newscast begins or ends with a progressive transition, as a dissolve, as illustrated by Figure 17.

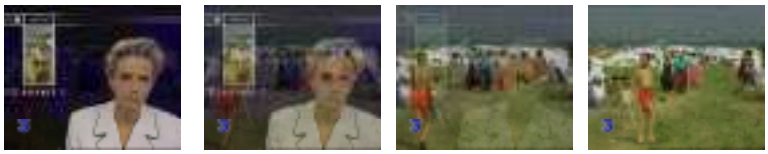


Figure 17: progressive transition at the beginning of a "Soir 3" report

Shots of the anchorman and progressive transitions of a typical "Soir 3" edition is illustrated by Figure 18. As progressive transitions may appear during a report, and as reports don't always start or end with a progressive transition, recognising reports with using only progressive transitions will produce some errors. These errors may come from a shot of the anchorman or a sequence of shots of the anchorman

[5] Still from AIM corpus

being classified as a report if it is surrounded by progressive transitions, or from "missed" reports – or more precisely missed transitions between two consecutive reports – caused by an anchorman shot or a sequence of anchorman shots neither starting nor ending by a progressive transitions.

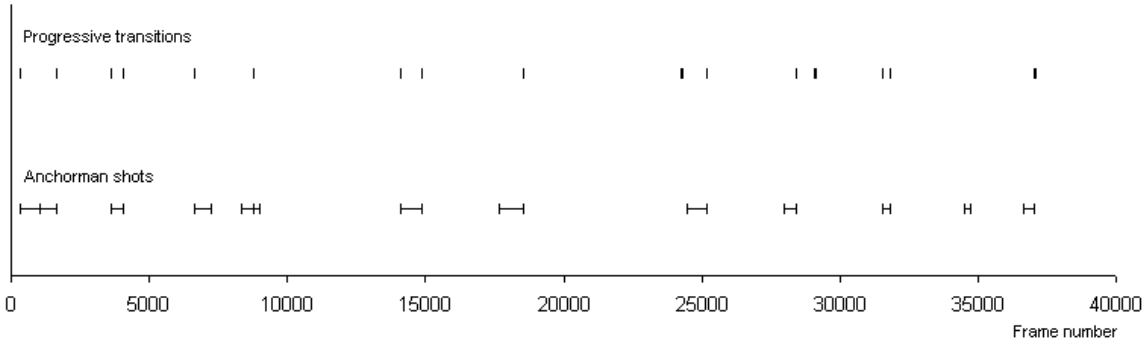


Figure 18: anchorman shots and progressive transitions in a "Soir 3" edition

The template used to recognize reports using the *ProgressiveTransition* event concept is the same as the template used with the *AnchorManShot* concept, by replacing *AnchorManShot* with *ProgressiveTransition*. Table 1 summarizes the number of reports recognized with the two templates, as the number of anchormans shots or sequences classified as reports and the number of missed reports.

Reports from anchorman shots	**Reports from progressive transitions**	**Anchorman shots or sequences classified as reports**	**Missed reports**
12	15	3	1
11	24	11	0
10	17	6	1
13	26	4	1
10	31	6	0

Table 1: reports from "Soir 3" editions

This experience shows that detecting reports in this case by using only the progressive transitions entails some over-segmentation due to progressive transitions appearing during the reports and anchorman shots or sequences of shots surrounded by progressive transitions. The quasi systematic use of progressive transitions at the beginning or at the end of reports leads to a very small number of forgotten reports.

Conclusion

We have shown in this paper that a plan recognition approach, making use of complex temporal relations between video segments can be both useful and efficient for solving a variety of video segmentation and indexing tasks. This temporal framework is obviously not sufficient for solving all problems, and should be extended to deal with other relations, such as image or sound similarity

between segment classes, and numerical temporal constraints. While this paper focused on the macro-segmentation task, such extensions could be even more useful in other applications, such as the automatic generation of video abstracts. More work is also needed to determine whether this temporal framework is necessary (compared to other simpler approaches using regular expressions and finite automata) for solving the task at hand.

This work should be extended in two main directions. First, experimentations showed that designing a template for a collection is not a trivial task, even for television experts, and that the quality of the results depends critically on such difficult design choices as temporal constraints. Therefore, we are now turning to machine learning techniques for creating templates from annotated examples. Second, we currently assume that the initial segmentation and classifying results observed by Clavis are perfect. In the future, we would like to relax this assumption, for instance by defining preferences in the space of solution of the CSPs.

References

Aigrain, P., Joly, P., Longueville, V. (1997). Medium Knowledge-Based Macro-Segmentation of Video into Sequences. Intelligent Multimedia Information Retrieval. A. P. M. Press.

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM* 26, 832-843.

Borgida, A., Brachman, R.J., McGuiness, D.L., Resnick, L.A. (1989). CLASSIC: A Structural Data Model for Objects. *ACM SIGMOD Int. Conf. on Management of Data* (pp. 59-67)

Bouthemy, P.& Garcia, C.& Ronfard, R.& Tziritas, G.& Veneau, E. & Zugaj, D. (1999). Scene segmentation and image feature extraction for video indexing and retrieval. *Third International Conference on Visual Information Systems (VISUAL'99)*, Amsterdam, june 1999

Carrive, J.& Pachet, F. & Ronfard, R. (1998). Using Description Logics for Indexing Audiovisual Documents. *Proceedings of the International Workshop on Description Logics*, Trento, Italy

Drakengren, T. & Jonsson, P. (1997). Twenty-one Large Tractable Subclasses of Allen's Algebra. *Artificial Intelligence* 93, 297-319.

Fontaine, D. (1996). Une approche par graphes pour la reconnaissance de scénarios temporels. *Revue d'Intelligence Artificielle* 10(4), 439-468.

Kautz, H. A. (1991). A Formal Theory of Plan Recognition and its Implementation. Reasoning about Plans. J. F. Allen& H. A. Kautz& R. N. Pelavin & J. D. Tenenberg, Morgan Kaufman Plubishers, Inc.

Nebel, B. (1990). Reasoning and Revision in Hybrid Representation Systems. *LNAI* 422.

Nebel, B. & Bückert, H. J. (1994). Reasoning about Temporal Relations: A Maximal Tractable Subclass of Allen's Interval Algebra. *Proceedings of AAAI'94*, Seattle, Washington (pp. 356-361)

Ramaux, N. & Fontaine, D. (1996). Recognising a Scenario by Calculating a Temporal Proximity Index between Constraint Graphs. *8th International Conference on Tools with Artificial Intelligence (ICTAI 96)*

Ronfard, R. (1997). Shot-level description and matching of video content. *SPIE 97*, San Diego

Roy, P.& Liret, A. & Pachet, F. (1999). The Framework Approach for Constraint Satisfaction. *Object Oriented Application Frameworks, Wiley Eds.* 2.

Thompson, R. (1998). *Grammar of the shot*, Focal Press.

Vilain, M.& Kautz, H. & Van Beek, P. (1989). Constraint propagation algorithms for temporal reasoning: A revised report. *Readings in Qualitative Reasoning about Physical Systems*, 373-381.

Weida, R. (1995). Knowledge Representation for Plan Recognition. *IJCAI 95 Workshop on the Next Generation of Plan Recognition Systems*, Montréal, Québec, Canada