Expert Systems with Applications 30 (2006) 694–704

Expert Systems with Applications

www.elsevier.com/locate/eswa

Automated cinematic reasoning about camera behavior

Doron Friedman[a], Yishai A. Feldman[b,*]

[a]VECG Laboratory, Department of Computer Science, University College London, UK
[b]Efi Arazi School of Computer Science, The Interdisciplinary Center, P.O. Box 167, 46150 Herzliya, Israel

Abstract

Automated control of a virtual camera is useful for both linear animation and interactive virtual environments. It has been partially addressed in the past by numeric constraint optimization and by idiom-based approaches. We have constructed a knowledge-based system that allows users to experiment with various cinematic genres and view the results in the form of animated 3D movies. We have followed a knowledge acquisition process converting domain expert principles into declarative rules, and our system uses non-monotonic reasoning in order to support absolute rules, default rules, and arbitrary user choices. We evaluated the tool by generating various movies and showing some of the results to a group of expert viewers.
© 2005 Elsevier Ltd. All rights reserved.

Keywords: Virtual camera; Knowledge-based systems; Virtual cinematography

1. Introduction

Automating camera behavior has applications for both off-line generation of linear animation and non-immersive[1] interactive virtual environments. For linear animation, automated camera control is an essential part of any fully automated tool for animation generation, and may be a useful component in high-level authoring tools. For interactive environments, the need is even more obvious: since a real-time director is not available, all camera-related decisions need to be done automatically.

Cinematographers have been dealing with camera issues for over a century. The cinematic expression is an arsenal of principles and conventions, including, but not limited to, camera behavior. Professional filmmakers learn these principles, while naive viewers who watch TV or film are usually not consciously aware of them. For example, a shot is a consecutive set of frames shot continuously by a single camera; shots are separated by cuts. Normally, a cut would be a startling event, where the point of view changes instantly. However, over the years filmmakers discovered how to use cuts, making them cognitively unnoticed (in the process, audiences were also 'educated' in the cinematic language).

These principles can be formalized. For example, 'jump cuts' are usually forbidden. These are cuts in which the camera's position does not change noticeably (an extreme case is where the camera stops and restarts from the same position after some time has passed). The rule usually found in film textbooks requires at least 30° difference in camera angle between shots. Sometimes, filmmakers deliberately violate these principles. For example, Godard introduced jump cuts in his film Breathless. It is this complexity that we want to capture; we want to be able to formalize rules, but also to allow the system to override them in specific circumstances.

In this paper, we attempt to formalize a non-trivial portion of the cinematic expression. We have used a knowledge-based approach, in which rules and principles were collected from textbooks and from interviews with a domain expert. In order to evaluate our approach, we have implemented a system called Mario. Our requirements from the system include flexibility, high-level knowledge specification, scalability, generality, and tractability. Mario is able to convert screenplays, given in a high-level formal language, into shooting scripts, which contain

* Corresponding author. Tel.: +972 99527305; fax: +972 99527246.
E-mail addresses: d.friedman@cs.ucl.ac.uk (D. Friedman), yishai@idc.ac.il (Y.A. Feldman).
URLs: http://www.cs.ucl.ac.uk/staff/d.friedman (D. Friedman), http://www.idc.ac.il/yishai (Y.A. Feldman).

[1] We assume that in immersive virtual reality the viewpoint is always identical to the participant's first-person view, although this convention may also be questioned and investigated.

0957-4174/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2005.07.027

explicit instructions on camera placement and behavior. Mario also includes a component that can generate an animation sequence from the script, if all the objects and actions included in the script are in the 3D library. The camera behavior in the output movie is determined by an automated reasoning process using a cinematic knowledge base.[2]

Clearly, an 'automated Fellini' is far beyond our capabilities, and thus our goals are more moderate; we hope that Mario's results can be useful for rapid prototyping and for specific applications. We have evaluated Mario with several example scenes from several TV genres. Each genre is reflected in a different knowledge base, and the resulting camera behavior is different. We show that even in the idiomatic genre of telenovela the reasoning can be complex. We have performed a preliminary evaluation of the results by showing the results to both experienced filmmakers and naive viewers.

2. Background

A virtual camera has at least seven degrees of freedom per frame: three for position, three for orientation, and one for field of view. Movies typically include 24–30 frames per second, so the search space for solutions is very large. Many types of considerations are involved: cognitive, aesthetic, and cultural. While the problem has been addressed by several researchers, we believe most would agree that it is far from solved.

Previous research into automated camera control can be classified into two methodologies: constraint satisfaction and an idiom-based approach. Constraint satisfaction methods (Drucker & Zeltzer, 1994; Drucker & Zeltzer, 1995; Bares & Lester, 1999; Halper & Olivier, 2000) typically work at the level of a single frame. Given a set of constraints about the objects to appear in the frame, they try to find the camera parameters that best satisfy the constraints. When such research is extended beyond a single frame, it no longer benefits from constraint optimizations (such as Drucker's camera modules (Drucker & Zeltzer, 1994)).

Idiom-based approaches (Karp & Feiner, 1993; Butz, 1997; Christianson, Anderson, He, Salesin, Weld, & Cohen, 1996; He, Cohen, & Salesin, 1996; Tomlinson, Blumberg, & Nain 2000) try to capture cinematic principles and let them dictate camera behavior. Cinematic principles may serve to reduce the large search space induced by the many degrees of freedom for a camera per frame. Several researchers have attempted some formalization of cinematic principles, and the abstraction of the space allows the use of symbolic algorithms rather than numeric methods.

The first to attempt some automatic cinematography were Karp and Feiner (1993); their ESPLANADE system used planning with communicative goals. Butz (1997) proposed that the rules of cinematic expression are analogous to grammar in natural language, and this was the basis for his CATHI system. Christianson et al. (Christianson et al., 1996) defined Declarative Camera Control Language (DCCL) and attempted a more systematic analysis of cinematography. They describe several cinematic principles and show how they can be formalized in a declarative language. They encode 16 idioms, at a level of abstraction similar to the way they would be described in a film textbook.

Real-time camera behavior was addressed by several projects. The Virtual Cinematographer (He et al., 1996) formulates some idioms as finite-state machines, which may then be used to make real-time decisions in 3D chat environments on the Web. Bares et al. investigated user modeling (Bares & Lester, 1997) and task sensitive camera behavior (Bares, Zettlemoyer, Rodriguez, & Lester, 1998). Tomlinson et al. (2000) used a behavior-based approach; the camera is modeled as an autonomous agent, and its behavior is based on a reactive behavior system, with sensors, emotions, motivations, and action–selection mechanisms.

Constraint-based approaches are more flexible and can be used to cover a wide range of situations. Idiom-based methods, on the other hand, have the advantages of abstraction and exploitation of domain knowledge.

Our approach is knowledge-based, and is thus more similar to the idiom-based approaches. However, we claim that idioms are the wrong granularity, being too coarse to formalize cinematic knowledge. Using idioms, it is necessary to code a specific idiom for every possible situation. This method results in a repetitive and predictable output, which impedes user engagement. For the same reasons, using idioms does not scale to cover a large variety of complex situations. Our analysis reveals a lower level of rules. Such rules may sometimes be combined together to form idioms, but the range of phenomena they cover is much larger due to their interaction. Another problem with idiom-based approaches is that they assume that an editing algorithm is known. We believe this view is too optimistic for a domain as complex as cinematography, and have thus opted for an automated reasoning approach.

3. Zooming in on Mario

3.1. Architecture and representation

Mario's architecture is shown in Fig. 1. The inputs to Mario are a screenplay and a floor plan. The screenplay is given in a formal language. The reasoning engine applies cinematic principles, taken from the cinematic knowledge base, to the inputs. The output is a list of the camera

[2] To view example animation files created by Mario follow the instructions in http://www.cs.ucl.ac.uk/staff/d.friedman/kbc.html.

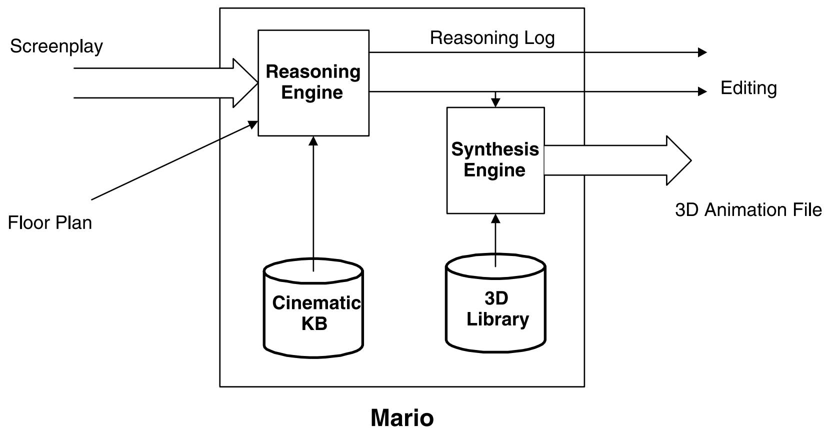

Fig. 1. Mario system diagram.

parameter values, as determined by the reasoning process. The reasoning engine also produces a log of its reasoning process. The system can create a synthetic 3D movie corresponding to the screenplay and the camera decisions, if 3D models and animations for the objects and actions mentioned in the screenplay are available.

The data representations typically used for computer graphics are not adequate for our needs; we need a scene description on a higher level of abstraction. We represent a scene as a collection of actions placed on a time line. The time line is annotated with spatial information: the position and gaze direction of each object is given in the beginning of the scene and in every point in time in which it changes. We split the time line into a list of intervals. For every beginning or ending of an action we insert an interval boundary on the time line. This allows us to avoid continuous time, and use a discrete representation in the form of a list of time intervals.

The automated reasoning portion of Mario is implemented in Common Lisp, based on top of Cake (Rich & Feldman, 1992). Cake is a multi-layered reasoning system developed at MIT's Artificial Intelligence Laboratory in the late 1980s. Cake's architecture includes seven layers; the bottom six layers provide the following generic knowledge representation and automated reasoning facilities: truth maintenance (TMS), equality, pattern-directed invocation, types, algebra, and frames. In addition to these generic reasoning layers, Cake also has a top layer for reasoning about the plan calculus, which is a specialized formalism for software development. We have replaced this layer with Mario, which is a specialized layer for reasoning about cinematography.

We refer to the camera parameters in a single frame as a viewpoint. Viewpoints are generally associated with numeric values for position, direction, and field of view. Theoretically, there is an infinite number of possibilities with seven degrees of freedom per frame. Our approach is different, and it assumes that viewpoints are best determined in story space. Thus, the geometric space is abstracted to include a finite number of possible viewpoints, based on the physical arrangement of the scene and on several cinematic attributes.

To describe a viewpoint, it is often enough to refer to the target objects or actors, the angle in which they are displayed, and the frame (image) composition. Viewpoints are implemented as frames, and so we refer to their attributes as slots. Thus, our formalization includes three main slots that are enough to specify a virtual camera, and additional slots that may be used to specify less conventional viewpoints. The three major slots are:

- *Target*: an actor or an object appearing in the scene;
- *Shot type*: close up (CU), medium shot (MS), or long shot (LS); and
- *Profile angle*: front-L, front-R, 3/4L, 3/4R, or back.

Shot types and profile angles are cinematic concepts. A frontal shot means that the camera faces the actor. Actors gazing directly at the camera look unnatural, so actors are trained to look a little to the left or to the right of the camera. The result is front-L and front-R, which refer to the case where the camera is oriented 30° from the target's gaze vector, either to the left or to the right, respectively (see Fig. 2). By a 3/4 profile angle we refer to 3/4 of a right angle

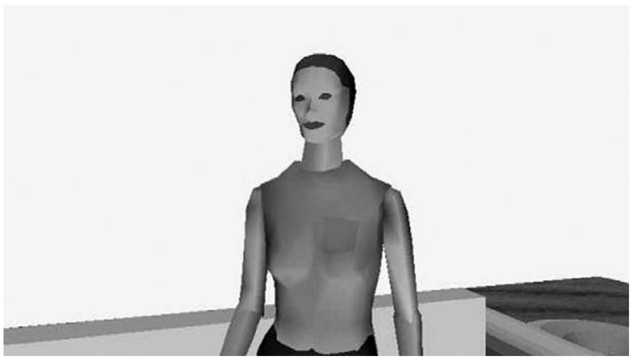

Fig. 2. Mother displayed from a frontal medium shot, i.e. with an angle of 30°.

(or 67.5°), either to the left or to the right of the target's gaze vector (see Fig. 3).

Using discrete values makes it possible to apply symbolic reasoning, allowing the computation to be simpler and more efficient. To support less conventional frame compositions, viewpoints have three additional slots: pitch, tilt, and zoom. It is also possible to add more discrete values to the three main slots; for example, cinematographers also define extreme close ups, extreme long shots, and other basic shot types; these can be easily introduced into Mario.

3.2. Cinematic rules

For the sake of the explanation, we will use a small scene fragment from the Latin telenovela *Dulce Anna*. We will illustrate how the non-monotonic interaction among low level cinematic rules gives rise to higher level cinematic idioms as well as to cinematic decisions that are valid yet were not preconceived by the domain expert. The rules and their formalization will be simplified; more details appear in the first author's PhD thesis (Friedman, 2003).

Recall that the time line is divided into intervals. Every interval has two viewpoints, one at the beginning of the interval and the other at the end; these viewpoints are accessed by the functions first-vp and last-vp, respectively. This formalization allows introducing additional viewpoints in the middle of intervals, but this was not utilized so far.

Assume that only the following rules in our knowledge base affect the situation (these are taken from the rules our domain expert formulated for the telenovela genre):

(1) If an actor is speaking, she is displayed in a frontal (30°) medium shot.
(2) If an actor is walking, she is displayed in a long shot, from a 3/4 angle.
(3) Cameras do not cross the line of interest with a cut (Every scene has a line of interest; for example, if there are two actors in the scene, the line of interest is the line that connects their two-dimensional positions. When a camera crosses the line of interest, the relative positions of the actors change, which is confusing if done abruptly. Crossing the line during camera motion is not prohibited, because viewers see the change as it occurs).
(4) There are no jump cuts; specifically, the angle between two consecutive shots is at least 60°.[3]

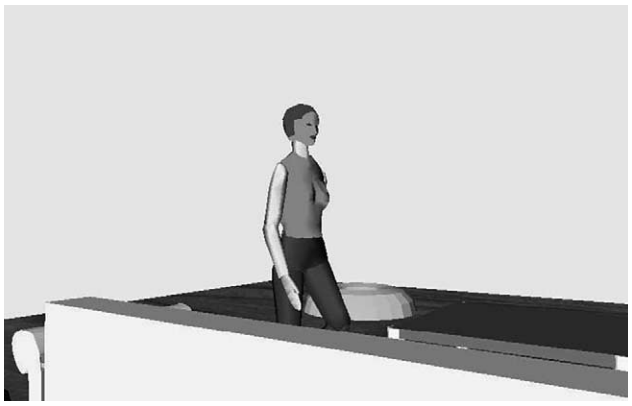

Fig. 3. Mother displayed from a long shot, 3/4 profile angle.

Note that these rules have lower granularity then cinematic idioms. For example, the first, third, and fourth rules given above, together, constitute the dialog idiom as it is mentioned in cinematic texts, and as it is used by idiom-based approaches such as the virtual cinematographer (He et al., 1996). In the sequel, we will see that these rules can also cover situations for which there is no consensual cinematic idiom.

The way we formalize the first rule is as follows: Mario checks for each interval I whether actor a speaks in that interval. If so, it adds the following axioms into Cake:

```
(= (target (first-vp I)) a)
(= (target (last-vp I)) a)
preference: (= (shot (first-vp I)) MS)
(= (shot (first-vp I)) (shot (last-vp I)))
(or (= (profile-angle (first-vp I)) front-L)
    (= (profile-angle (first-vp I)) front-R))
(= (profile-angle (first-vp I))
   (profile-angle (last-vp I)))
```

Note that these axioms are not quantified. They may appear many times in a scene, once for each interval in which an actor is speaking. Also note that one formula in the rule definition above is marked to be a preference. This is an example of a retractable formula; these will be explained in Section 3.3.

The formalization of the second rule is similar to the first. In this case, the formula assigning the value to the profile-angle is a retractable preference.

Next, we want to formulate the rule that states that the camera does not cross the line of interest with a cut. Note that until the basic slot values of the viewpoints are determined the viewpoint position is undetermined, and it is impossible to detect that the line is crossed. Cake allows this type of reasoning with the pattern-directed invocation mechanism, by installing demons that are triggered upon the creation of a new term.

As a first step, we install an axiom for every candidate cut point. Recall that only interval edges are candidates for cuts. For every interval I followed by an interval J, we install the following axiom, which requires the camera in the end of interval I and the camera in the beginning of interval J to be on the same side:

(same-side (last-vp I) (first-vp J))

[3] While cinematographers often cite 30° as the right number, in telenovela, being more conservative, differences smaller than 60° are rare.

We install a demon that is triggered whenever the pattern for the same-side function appears in a new term. This demon installs the following axiom:

```
(iff (same-side v1 v2)
     (= (line-side (shot v1)
                   (target v1)
                   (profile-angle v1))
        (line-side (shot v2)
                   (target v2)
                   (profile-angle v2))))
```

Next, line-side is assigned meaning, by another demon. It tests that all viewpoint values are initialized, and if so it computes the side of the line of interest on which the viewpoint is positioned. The geometric computation is thus carried out only when it is needed, and geometry can be kept out of the symbolic non-monotonic reasoning.

If the viewpoint is indeed determined, then this demon installs the following axiom:

(=(line-side shot target profile) *side-x*)

where *side-x* is one of the symbols side1 or side2, denoting either one side of the line of interest or the other.

The last rule in our example is the rule that requires at least a 60° angle difference between consecutive shots. The formalization of this rule in Cake is similar to that of the line-of-interest rule. For every interval I followed by an interval J, we install the following axiom:

```
(or (= (first-vp J) (last-vp I))     ; no cut
    (and (<= (abs (- (angle (first-vp J))
                     (angle (last-vp I)))
             (/ PI 3)))
         (<= (abs (- (angle (first-vp J))
                     (angle (last-vp I)))
             (/ (* 5 PI) 3)))))
```

where angle is a function that returns the 2D angle for a given viewpoint. The angle values in our computations are always between 0 and 2π, so this rule states that either the two viewpoints are the same, or that they have an angle difference greater than 60° ($\pi/3$). We assume a 2D representation and that the viewpoint direction is parallel to the $<x, z>$ plane.

Note the use of equality, which is supported by Cake. In this case, the equality is tested for two frame objects. So in fact, we are using the following equation, automatically provided by Cake:

```
(iff (= v1 v2)
     (and (= (target v1) (target v2))
          (= (shot v1) (shot v2))
          (= (profile-angle v1) (profile-angle v2))
          (= (tilt v1) (tilt v2))
          (= (zoom v1) (zoom v2))
          (= (pitch v1) (pitch v2))))
```

A demon is triggered when an angle term is detected. As in the case of the line-side computation, there is no point in trying to compute the angle before the relevant slots are assigned values. In this case, only the viewpoint direction is necessary, so we only need the target and the profile angle to be determined. When they are determined, the demon computes the rotation angle dictated by the slot values.

This concludes the formalization of the four rules. Next, we turn to describe the reasoning process. We will illustrate the process with a very simple example, in which the actions do not overlap. There are two actors, the teenage Mario and his mother. The input description of the scene is:

0 3 Mother walk-to point-1
3 6 Mother speak 'Have you eaten the sandwich I prepared for you?'
6 8 Mario speak 'I wasn't hungry.'

Even in this simple example, we can see an interaction between the rules, which yields an editing solution that may not have been anticipated by the domain expert.

This scene fragment includes three intervals: [0,3], [3,6], and [6,8]. Since Mario speaks in the third interval, he should be shown in a frontal medium shot (first rule). Because he is not looking at his mother, but towards the center of the living room, both directions (front-left or front-right) are on the same side of the line of interest. Therefore, when Mother speaks in the second interval, she should be shown with a frontal medium shot (rule 1), from the same side of the line (rule 3). In the first interval, Mother walks, and so should be shown with a long shot from a 3/4 angle (rule 2), still from the same side of the line (rule 3).

This, however, is impossible, since it results in a 'jump-cut' between the first and the second intervals, which is a violation of the fourth rule that requires at least a 60° difference between consecutive shots. The reason is that the difference between a 3/4 angle and a frontal shot (30°) is 37½°, less than the required minimum of 60°.

Now the system needs to resolve this conflict, and it is able to do so, since some of the constraints were formulated as preferences. In this example, the rules required a simple shot, i.e. that both viewpoints for the same interval will be equal. There is only one way to satisfy all constraints, which is by unifying the first two intervals into one shot, in which all viewpoints will display a frontal long shot of Mother. This is a new kind of shot, which was not explicitly mentioned in any of the rules, and was not anticipated by the domain expert. This demonstrates a type of emergent behavior, which is highly desirable in entertainment and artistic contexts: the expert specified rules with one scenario in mind, and the rules interacted in a meaningful way, to generate an unexpected result for a new scenario. If the rules were formulated correctly, this would form an acceptable solution. This interaction of rules allows for the kind of scalability that does not exist in the idiom-based approach.

3.3. Non-monotonic reasoning

Cake provides different kinds of retractable premises, with different behaviors. In our implementation of Mario, we have used two of the premise types provided by Cake: defaults and assumptions, and have added a third mechanism into Cake, which we call preferences.

The differences between these different types of premises are as follows. **Defaults** are assumed to be true, and they are the first to be retracted in the case of a contradiction. If, during the reasoning process, the reason for retracting a default is by itself retracted, the default would automatically pop back and be assumed true. For example, in a telenovela, we would probably use a default to specify that the camera is stationary. **Assumptions** are assumed true, but unlike defaults, they do not get retracted or popped back automatically by Cake. We typically use assumptions to represent arbitrary choices that result from disjunctive clauses. For example, the decision about side of the line-of-interest would be formalized with an assumption, since using either side is arbitrary. If it turns out that the original, arbitrary, choice leads to a contradiction; it is automatically retracted and replaced by the alternative choice.

During the development of Mario, it turned out that these two mechanisms were not enough. We have introduced **preferences**: these are similar to defaults in that they are automatically popped back if the reason for their discarding is retracted. However, unlike defaults, they are the last to be retracted. As an example for the need for preferences, consider a situation in which we want one of the actors to be displayed in a close up, if possible. Such a premise can interact non-monotonically with other premises, and we would like it to hold whenever possible.

The following algorithm for resolving contradictions uses the function $m(a)=\min\{intervals(a)\}$, where *intervals*$(a)$ is the set of indexes of intervals referred to in the premise a.

```
$C \leftarrow$ set of premises involved in contradiction
**if** $C$ is empty
**then fail** 'cannot resolve contradiction'
**let** $D=\{d\in C | d$ is a default$\}$
    $A=\{a\in C | a$ is an assumption$\}$
    $P=\{p\in C | p$ is a preference$\}$
**if** $D$ is non-empty **then** retract $d$ s.t. $m(d)$ is maximal in $D$
**else if** $A$ is non-empty **then** retract $a$ s.t. $m(a)$ is maximal in $A$
**else if** $P$ is non-empty **then** retract $p$ s.t. $m(p)$ is maximal in $P$
```

As explained above, we see that in case of contradictions, Mario first tries to retract defaults. If no defaults are involved it tries to retract an assumption.[4] A preference will only be retracted if the contradiction includes neither defaults nor assumptions.

In the first author's PhD thesis (Friedman, 2003) we show that the algorithm terminates, although it is potentially exponential. Choosing the premise that refers to the intervals with the greatest indexes is a heuristic that dramatically improves the performance of the algorithm. Most rules refer to a single interval, or to two adjacent intervals. The latter cause dependencies between intervals, which can propagate along the whole timeline. For example, the arbitrary choice of which side of the line-of-interest to use could propagate from the first interval until the last. Reasoning usually proceeds from the first interval forwards. If a contradiction is found in the last interval, and we choose to retract a premise that refers to the first one, the retraction is likely to propagate along all intervals, causing many retractions and necessitating many forward deductions to replace them. If, however, we retract a premise that only refers to the last two intervals, the effects are likely to be significantly smaller. The introduction of this heuristic reduced the run time for one example from several hours to less than two minutes.

4. Results

We have run Mario and evaluated the results for over a hundred scenes. Most of the scenes only involved two actors, and some included three actors. Due to limitations in animation generation, most of our examples only included a restricted set of actions: walking, speaking, jumping, and running. The largest example we analyzed was the complete *Dulce Anna* scene, which included 41 actions over 45 intervals, resulting in an animation sequence of 2 min and 20 s.

The telenovela example was tested with a variety of rule sets. The version of the knowledge base that produced the best results, according to our domain expert, includes eight rules, each including several formulae. During the processing of the full scene, the TMS was loaded with over 10,000 terms.

Running Mario with the whole telenovela scene takes approximately 10 s on a PC with 2 GHz Pentium 4

[4] Mario retracts assumptions automatically, even though this is not the case in Cake.

processor, 1 GB of RAM, running GNU Common Lisp. This is acceptable for an off-line tool, which may be used interactively through iterative refinement of results.

In the rest of this section, we discuss some of the interesting issues that came up during the formalization.

4.1. Shots and establishing shots

Shots pose an interesting problem: they are a central concept in cinematic language, and thus rules need to refer to them. A shot is a consecutive set of intervals, but where shots start and end is not known in advance; part of the goal of the rules is to define them. Thus, we are faced with rules that refer to entities whose parameters are unknown. This is one of the main reasons for using non-monotonic reasoning. Our solution is to define shots using a function, shot-number, which maps an interval to the index of the shot it is part of. The values of this function can change non-monotonically throughout the reasoning process.

Establishing shots are used by filmmakers to make their viewers familiar with the location of the scene. This is typically a long shot, a very long shot, or even an extreme long shot, which displays the whole location for a duration that is long enough for viewers to absorb the spatial relationships between all significant objects in the scene. Naturally, it appears in the beginning of the scene, but it does not need to be the first shot. Also, if the spatial relations among the main actors and objects change significantly during the scene, such a shot is required again, and is called a *re-establishing shot*. A re-establishing shot may be different from an establishing shot; for example, it may be enough to provide a long shot of the objects that changed their position, rather than display the whole space again. However, in the presentation below we treat them in the same way.

Our domain expert provided two sets of rules: one for specifying when an establishing shot or a re-establishing shot is required, and another that defines the camera behavior for such shots. The problem introduced by this concept is that it applies to a shot, rather than to actions or intervals, so rules relying solely on intervals will not be adequate.

We can define establishing shots as a predicate, called ES, over shots. Whenever an expression such as (shot-number i) is created, we add (using a demon) a default that states that the shot is not an establishing shot, or:

(not (ES (shot-number i)))

This is the required infrastructure. Next we need a set of rules that specify a shot to be an establishing shot. First, we have the following rule stated by the domain expert: ‘The first or the second shot of a scene must be an establishing shot’. This is formalized using the following axiom:

(or (ES 0) (ES 1))

Another rule specifies a re-establishing shot: if the spatial configuration of the scene changes significantly at time t, and the interval that starts with t has index i, we add the formula:

(or (ES i) (ES (+i 1)))

In addition to the rules specifying which shots are to be establishing shots, we need rules to specify the camera behavior in such shots: for example, that they should be long shots. This needs to be installed for every interval, since whether a shot will be an establishing shot or not may change non-monotonically throughout the reasoning process. For every interval I we install the following formula:

```
(implies (ES (shot-number I))
          (and (= (shot (first-vp I)) LS)
                (= (shot (last-vp I)) LS)))
```

The mechanism described here guarantees that there is a long shot in at least one of the first two shots after a spatial change in the scene. Note that the shots themselves may change non-monotonically throughout the reasoning process.

4.2. Manual intervention

In Section 3.3, it was explained how Mario automatically resolves contradictions involving retractable formulae. User intervention in such cases is also possible; Cake allows querying the user which premise to retract. Cake also allows the user to trace the reasoning leading to a specific formula. The user can thus interact with Mario, follow Mario’s reasoning, and decide which premise to remove.

User intervention is not limited to the selection of premises to retract in the face of a contradiction. The user may view the results, and may decide to add a piece of information. For example, say that the user is unhappy with the solution that was derived automatically for the example described above. Perhaps the user knows something that the system could not deduce from the screenplay, which requires Mario to be displayed in close up in interval I. In such a case, it is possible to add rules expressing this new information, into the system, such as the following preferences:

```
(and (= (shot (first-vp I)) CU)
     (= (target (first-vp I)) Mario))
(= (first-vp I) (last-vp I))
```

This new information is then propagated in the TMS, and it may interact with other premises or rules. The user can determine the effects of her preferences, and be alerted if it violates any cinematic principle. Note that this does not require the user to change the cinematic knowledge base; it is enough to add a specific piece of

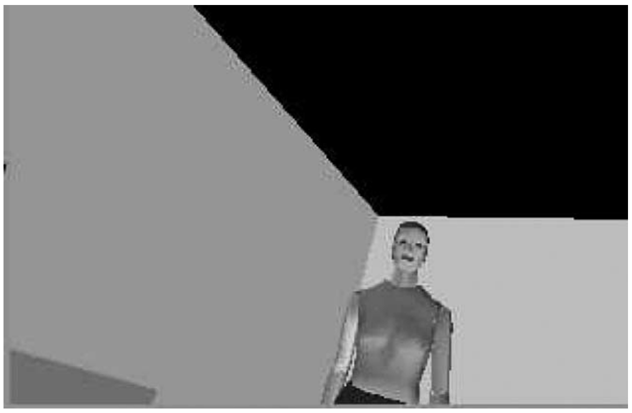
Fig. 4. If a character is perceived to be threatening, a rule can specify that the character be displayed from a low angle.

information that is only relevant in the context of the given screenplay.

4.3. Additional genres

In addition to Latin telenovela, we have analyzed scenes from other genres, mainly a more dramatic genre (from the TV science-fiction series The X-Files). This introduced new challenges; mainly, it includes a much richer set of situations, locations, and shot types compared to telenovela. For example, low-angle shots (Fig. 4) are sometimes used to portray an actor in a threatening attitude.

As we refined the knowledge base and introduced new rules, we frequently confronted the problem of contradictions between rules. It is possible to keep the problem under-constrained by using preferences and defaults rather than axioms, but this might result in too many arbitrary choices.

Conflicting defaults is a well-known problem in non-monotonic reasoning, but there does not seem to be a general solution. We have examined a domain-dependent solution to the problem of contradicting defaults: classifying rules by the high-level goals they are meant to achieve. We have identified several categories of such high-level goals: spatial orientation, conveying the information explicit in the script, conveying the information that may be implicitly deduced by a viewer from the script, aesthetic considerations, and parsimony: using the minimum number of shots and cameras. In cases of conflicts, the system can try to satisfy all the goals, rather than choose preferences arbitrarily. The goals component has not been implemented yet.

4.4. Experimental cinematography

The formalization may help the filmmaker organize the cinematic knowledge in her mind. This is useful not only for gaining new insight into cinematic expression; by abstracting and formalizing the domain space, the filmmaker may become aware of new options.

We can illustrate this by examining two examples of movies created by Mario, by using simple and deliberate violation of rules. These examples would have clearly been impossible using the idiom-based approach. In the first, we asked Mario to prefer complex shots over simple shots, by requiring, for every interval, that the first viewpoint will be different from the last. Note that our method is fully deterministic, which means that even if there are arbitrary choices, we expect a high degree of consistency. In the case described above, Mario preferred modifying the profile angle from 3/4 to frontal, and also preferred zooming in. The result was of a very consistent style including repeated camera rotations, which several viewers called the ‘Matrix’ style.

An additional small modification produced a completely different style. Instead of requiring, for each interval, that the first viewpoint will be different from the last one, we required that for each interval, **all** the three basic slots of the first viewpoint be different from the corresponding slots in the last viewpoint. We expected a very dynamic camera behavior, but watching the resulting movie was still surprising. The consequence of the new rules was that in almost all of the shots, the camera rotated between the two actors. This turned out to be a very consistent style, called by some viewers the ‘Ping Pong’ style. Some film students mentioned that this style reminded them of the Dogma 95 cinematic genre, which is characterized by unstable and rapidly changing camera positions. Filmmaker Yigal Burstein remarked that this style reminded him of an experimental film by filmmaker Michael Snow.[5] Snow used repetitive camera rotations in this experimental film in order to accentuate a sudden dramatic event. Thus, we see that, on the one hand, experimenting with the rules can lead to surprising effects. But on the other hand, we are far from a tool that would deliberately select such a style to emphasize an event, as done by Snow.

4.5. Empirical evaluation

We have conducted an informal evaluation of Mario, by presenting several output movies to an audience, and having the audience fill in a questionnaire. The experimental setting had several limitations, and the results need to be interpreted with caution, or verified by a systematic experiment.

The main idea was to conduct a kind of a Turing test, that is, to see if viewers, and especially filmmakers, could tell the difference between Mario’s editing and expert human editing. We have tested two groups. Group A was composed of 12 graduate film and TV students, who are assumed to be ‘native speakers’ of the cinematic language. Group B was composed of 10 people who had no background in film and TV.

[5] This film from 1968 was given no name, and is usually referred to as ‘Back and Forth’.

Both groups were given a simple questionnaire. In addition to the questionnaire, the subjects were provided with an oral explanation of what we mean by ‘human version’, i.e. the fact that the film expert made the editing decisions, and that the animation was generated automatically. Group A answered the questionnaire in class, while group B went through the experiment individually.

We used five versions of the telenovela scene in 3D animation:

(1) the best version generated by Mario, according to the telenovela knowledge-base;
(2) a version generated by Mario, with all the telenovela rules except the line-of-interest rule, which was removed;
(3) a version generated manually by our domain expert;
(4) a version generated by Mario, specifically generated to cause moderately dynamic camera behavior (the ‘Matrix’ style), as described in Section 4.4; and
(5) a version generated by Mario, specifically generated to cause extremely dynamic camera behavior (the ‘Ping Pong’ style), as described in Section 4.4.

We wanted to use another version similar to the one in the original TV series. However, due to limitations in animation generation, the animated scene is quite different from the original one, so this was not possible.

Mario’s editing was based on the principles formulated by the same domain expert who edited the manual version. So, in principle, the two versions should have been exactly identical. However, it turned out that there were some differences. When we interviewed the domain expert about his version, he admitted that he used additional rules and considerations, which he did not provide to Mario. One of these is a ‘voice over’ editing technique. For example, in part of the scene Mario speaks for 5 s, and then his mother provides a reply, which lasts for a significant duration. Instead of cutting after 5 s, our editor left the camera on Mario for 7 or 8 s, thus showing Mario while his mother was already speaking.

We could have further extended the Mario system to incorporate all the new principles, until its version was identical to the one generated by the domain expert. We decided to carry out the experiment in the way we did, because we thought it still reflects the difference between a knowledge-based system and a human expert; a knowledge-based system will always be some approximation of an expert.

After watching the movies, both groups of viewers were asked two questions:

(1) Do you think the editing decisions in the version you saw were made by a person or by a machine?
(2) How much did you like this version (on a scale of 1–5)?

For the first question, we only asked about the first three versions of the edited scene. The results are summarized in Table 1. The results of the rating tests are summarized in Table 2.

Table 1
The percentage of people who correctly passed the ‘Turing test’ by group, for the first three versions

	Telenovela	No-line	Domain expert	All three
% Correct in group A	67	67	50	33
% Correct in group B	60	70	60	40
% Correct in both groups	64	68	54	36

The last column shows the percentage of people who gave the correct results for all three versions.

In addition, the subjects were explicitly encouraged to comment on cinematic problems. All the film students provided comments, but only two (16%) noted the line-of-interest violations.

The following points can be made about the results. First, only 8 out of 22 (36%) of the viewers recognized who made all three movies correctly. Due to the limitations of the experiment, we only see this as a first indication that Mario’s result are comparable to human expert; this needs to be further investigated.

Group A was less successful in the ‘Turing test’: only 33% recognized all three movies correctly, compared to 40% of group B. Since data were not normally distributed, a Kruskal-Wallis two-way ANOVA test was performed on the two groups. No significant group differences have been found.

The overall rating results are interesting. First, we note that in both groups the rating for the human-made version was highest. This was statistically significant in related *T*-tests comparing the rating for the first and the third movie versions ($t=3.954$, $p=0.01$), and comparing the rating for the second and for the third movie versions ($t=3.356$, $p=0.03$). This is true even though, in general, the viewers were not aware that this version was created by a human. It seems that the question whether the movie was done by human or machine is very confusing, but that viewers do sense a difference, and prefer the human version.

An even more surprising result is that the rating of Mario’s correct telenovela version (first movie) was lower than the rating of the erroneous version (second movie). However, this was not statistically significant. Also, because the experiment was conducted in a classroom, we could not

Table 2
Grading movies edited by Mario and by the domain expert

	Rating by group A	Rating by group B	Combined rating
Telenovela	2.25	3	2.59
No-line	2.83	3	2.91
Domain expert	3.75	3.8	3.77
‘Matrix’	1.5	1.42	1.59
‘Ping Pong’	1.42	1.17	1.41

control the order in which the movies were displayed; this could have affected the results.

Almost all subjects rated the two 'experimental movies' very low. Perhaps there was a misunderstanding and they thought these were supposed to conform to the telenovela genre, and perhaps they were biased because they knew they were computer-made. One film student really liked the 'Ping Pong' version (graded it 5). Another film student liked the 'Matrix' version (graded it 4). We note that none of the film-illiterate group liked any of these versions. Both of the experimental versions ('Matrix' and 'Ping Pong') made an excessive use of one unique effect each. Viewers indeed comment that their was 'too much motion'. It would be interesting to have Mario generate versions that make a more subtle use of the effects, and see how viewers react to these.

5. Discussion and future work

Automated virtual camera is still an open problem, despite this research and others. Cinematic expression is a complex domain with a very large body of knowledge, and we do not claim to have covered it completely. In this section, we state some of the limitations of this work, and some of the ways in which we expect they can be overcome. We believe that this domain is a challenge that provides an opportunity for much more research in the future, specifically within the knowledge-based approach.

5.1. Limitations of the Mario implementation

Mario is now able to deal with a large number of cinematic concepts, and is able to perform high-quality decision-making about camera behavior. However, cinematic expression is a wide, possibly infinite domain. For example, Mario's frame compositions are based on a finite set of possibilities that are common in cinematography. Mario's frame decisions can thus be considered idiomatic, and thus have the disadvantages that we have associated with idiom-based approaches: it is not flexible enough to cover less typical situations, and does not allow evolution of new frame compositions. Numeric optimization approaches, such as Drucker's (Drucker & Zeltzer, 1994; Drucker & Zeltzer, 1995), are successful and appropriate for solving frame compositions based on constraints. It would have been interesting to integrate both methods; this is a candidate integration point.

In the lack of frame considerations, Mario also runs into the problem of occlusions. Since Mario relies on idiomatic frame compositions this would not be a problem in beginnings or ends of shots, but occlusion might occur during a shot if the actor positioning is non-conventional or if the actors are moving. There was one such occlusion in Mario's editing of the telenovela scene (Fig. 5). In this case, the domain expert mentioned that he liked the resulting effect, but stated that it would have probably been avoided in a telenovela.

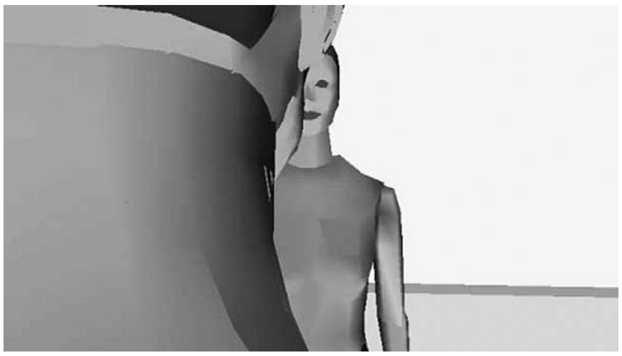
Fig. 5. Mother is partially occluded by Mario, during a shot with camera motion.

If occlusion computation is supported, this has another interesting implication: occlusions can be utilized to achieve interesting cinematic effects, as is done in advanced cinematic genres. One example is partially occluding a target for aesthetic reasons; this is also related to frame composition. Another example would be gradually exposing a surprising target to the viewer, by moving the camera to reveal it from behind an occluding object.

5.2. Manipulation of time

The conversion of a story or a screenplay into a movie typically includes temporal transformations. There are at least two relevant time lines: story time, and presentation time. An automated tool for creating visual movies from textual scene descriptions may want to apply various transformations to the story events: shorten, stretch, omit, add, or change the order of events.

Mario only makes camera-related decisions, but in a follow-up research (Friedman, Feldman, Shamir, & Dagan, 2004) we have experimented with a tool that creates automated summaries out of virtual environment sessions. This tool included manipulation of time: in order to generate summaries it had to automatically remove events from the story.

It is clear that we cannot just remove subsets of the story; continuity must be preserved. While we have only experimented with one simple type of temporal transformation, we have learned a few lessons. First, we were careful in deciding what points to cut. If a certain sub-scene is interesting, we may want to include a few events earlier that illustrate how the spatial situation was constructed. Also, if significant changes took place during a sub-scene we decide to remove, some of these changes need to be displayed. For example, if the actors in the scene significantly change their position, they need to be displayed walking in to their new positions. This may include displaying subsets of actions, or in extreme cases make up actions that did not really take place. Further research into temporal manipulation is required.

5.3. Automation

There is frequently a tension between human authoring and automation. Human authoring still offers maximum control and highest quality, and automation is used for minimizing labor. Recently, however, animators are learning that automation does not only save time, but can also allow them to focus on the important details, or even create animations that would have been impossible to create manually. We observe that the balance is shifting towards increasing automation, while allowing the humans to maintain control by intervening in certain points throughout the process.

Complete automation is difficult, especially when human-level AI is required; it does not seem that such AI will be around anytime soon. Reverting to semi-automated processes is therefore necessary, but this comes with a price; if some user intervention is required, the tool needs to provide facilities for man-machine collaboration, such as explanation capabilities.

With Mario we have taken the knowledge-based approach, which allows human users into the process in several stages: formalizing the principles, understanding the results generated automatically, and having the possibility to intervene in specific points. We believe such an approach is necessary in such a difficult domain, at least until automation procedures become widely accepted.

6. Conclusion

While our goal was to come up with a generic method for automated camera control, we have only looked at a few TV genres. We expect our method to extend to other domains, but this needs to be investigated. We have started to apply 'automated cinematography' in the context of manufacturing simulations. It now seems that the method could be very similar to the one described in this paper, with the main differences being the 'cinematic language' used in manufacturing simulations: occlusion plays a much more important role, special views, such as wire frame and cross-section cuts, are occasionally used, and events often take place in parallel.

We conclude by noting that recent advances in computer graphics, together with the growth of available processing power, stress the need for automation in animation generation, and allow the generation of more sophisticated virtual environments. It may be worthwhile to revisit old artificial intelligence techniques, as well as new ones, and see how they can allow us to achieve these goals.

Acknowledgements

We would like to thank Noam Knoller, our domain expert, and Amiram Yehudai for his support throughout the project.

References

Bares, W. H., & Lester, J. C. (1997). Cinematographic user models for automated realtime camera control in dynamic 3D environments. In A. Jameson, C. Paris, & C. Tasso (Eds.), *Proceedings of Sixth Int'l Conference User Modeling (UM97), Sardinia, Italy*, 215–226.

Bares, W. H., & Lester, J. C. (1999). Intelligent multi-shot 3D visualization interfaces. *Knowledge-Based Systems*, *12*(8), 403–412.

Bares W. H., Zettlemoyer L. S., Rodriguez D., & Lester J. (1998). Task-sensitive cinematography interfaces for interactive 3D learning environments. In *Proceedings of 1998 International Conference on Intelligent User Interfaces (IUI-98), San Francisco, California* (pp. 81–88).

Butz, A. (1997). Animation with CATHI. In *Ninth Innovative Applications of Artificial Intelligence (IAAI-97), Providence, Rhode Island* (pp. 957–962).

Christianson, D., Anderson, S., He, L., Salesin, D., Weld, D., & Cohen, M. (1996). Declarative camera control for automatic cinematography. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence, Portland, Oregon* (pp. 148–155).

Drucker, S. M., & Zeltzer, D. (1994). Intelligent camera control in a virtual environment. In *Proceedings of Graphics Interface '94*.

Drucker, S. M., & Zeltzer, D. (1995). Camdroid: A system for intelligent camera control. In *SIGGRAPH Symposium on Interactive 3D Graphics*.

Friedman, D. (Oct. 2003). Knowledge-based cinematography and its application to animation, PhD thesis, Tel Aviv University.

Friedman, D., Feldman, Y., Shamir, A., & Dagan, T. (2004). Automated creation of movie summaries in interactive virtual environments. In *Proceedings of IEEE Virtual Reality 2004, Chicago, IL* (pp. 191–198).

Halper, N., & Olivier, P. (2000). CAMPLAN: A camera planning agent. In *AAAI 2000 Spring Symposium on Smart Graphics, Stanford* (pp. 92–100).

He, L., Cohen, M. F., & Salesin, D. H. (1996). The virtual cinematographer: A paradigm for automatic real-time camera control and directing. *Computer Graphics*, *30*, 217–224.

Karp, P., & Feiner, S. (1993). Automated presentation planning of animation using task decomposition with heuristic reasoning. In *Proceedings of Graphics Interface '93, Toronto, Canada* (pp. 118–127).

Rich, C., & Feldman, Y. A. (1992). Seven layers of knowledge representation and reasoning in support of software development. *IEEE Transactions on Software Engineering*, *18*(6), 451–469.

Tomlinson, B., Blumberg, B., & Nain, D. (2000). Expressive autonomous cinematography for interactive virtual environments. In *Proceedings of Fourth International Conference Autonomous Agents, Barcelona, Spain* (pp. 317–324).