Knowledge-Based Cinematography and its Applications

Doron Friedman[1] and **Yishai A. Feldman**[2]

Abstract.
Automated control of a virtual camera is useful for both linear animation and interactive virtual environments. 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.

1 Introduction

Advanced virtual environments and artificial intelligence can be highly synergetic. We often want the environments to be intelligent, and to be inhabited by intelligent virtual creatures. One of the interesting challenges in 3D environments, which has been mostly overlooked, is virtual camera behavior. The camera is a critical tool for conveying information and affecting user experience, but its behavior is difficult to specify.

Cinematographers have been dealing with camera issues for over a century. Cinematic expression includes many principles and conventions, although naive viewers who watch TV or film are usually not aware of them. For example, "jump cuts" are typically avoided by mainstream filmmakers [27]. Avoiding jump cuts can be formulated as follows: after a "cut" (camera stops and re-starts shooting), the camera angle must change by at least 30 degrees. Such rules can be captured mathematically, and can thus be simulated by software. However, jump cuts were deliberately introduced into the cinematic language by French filmmaker Godard. 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.

Automating camera behavior has applications for both off-line generation of linear animation and real-time 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.

Our main goal in this research is to cover a significant portion of the cinematic expression; therefore, we stress the importance of the knowledge-acquisition phase. We have used a knowledge-based approach, in which rules and principles were collected from textbooks [6, 25, 27] and from interviews with a domain expert.

In order to evaluate our approach we have implemented a system called Mario.[3] 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 3D animated movies. The camera behavior in the output movie is determined by an automated reasoning process using a cinematic knowledge base.[4]

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 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: for position, orientation, and 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 idiom-based reasoning. Constraint-satisfaction methods [2, 9, 10, 17] 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.

Idiom-based approaches 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 works 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 [19]; their ESPLANADE system used planning with communicative goals. Butz [5] 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. [7] defined DCCL (Declarative Camera Control Language) 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.

[1] Department of Computer Science, University College London, d.friedman@cs.ucl.ac.uk. This author was partially funded by the European Union project PRESENCIA, IST-2001-37927.
[2] Efi Arazi School of Computer Science, The Interdisciplinary Center, Herzliya, Israel, yishai@idc.ac.il

[3] The system is named after one of the main characters in the telenovela that was selected as the first test case.
[4] To view example animation files created by Mario follow the instructions on: http://www.cs.ucl.ac.uk/staff/d.friedman/kbc.html.

Real-time camera behavior was addressed by several projects, The Virtual Cinematographer [18] 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 modelling [1] and task sensitive camera behavior [3]. Tomlinson, Blumberg, and Nain [26] used a behavior-based approach; the camera is modelled as an autonomous agent, and its behavior is based on a reactive behavior system, with sensors, emotions, motivations, and action-selection mechanisms.

Automated camera control has also been investigated in contexts other than virtual environments. Gleicher and Masanz [16] and Liu et al. [20] describe systems for automated video recording and editing. Nam and Thalmann [23] deal with an automatic camera in the context of a virtual theater, where human participants' bodies are tracked and projected into the animated scene. Bowers [4] describes Inhabited Television, which combines collaborative virtual environments (CVE) and broadcast TV; this is another innovative application that requires automated camera control.

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 an artificial intelligence approach, such as 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, one needs 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 more complex situations. Our approach looks at a lower level of facts, which together comprise idioms. 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 a declarative knowledge representation approach.

3 Zooming In on Mario

3.1 Architecture and Representation

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 parameter values, according to the knowledge base. 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.

We represent a scene as a collection of actions placed on a time line, with some geometric information attached. 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 a continuous time line, and use a discrete representation in the form of a list of time intervals; the advantages will become clear below.

We translated the domain expert's principles into formal rules. Initially, we used an ad-hoc reasoning algorithm, based on constraint propagation. However, during the evaluation of this system, the need for truth maintenance, and specifically the need for dependency-directed backtracking [21], became evident.

We have thus decided to rewrite Mario based on an existing system, implemented in Common Lisp, called Cake [24]. 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) [8, 21], equality, pattern-directed invocation [11], types, algebra, and frames [22]. In addition to the generic reasoning layers, Cake also has a top layer called *Plan Calculus*, which includes generic facilities to implement 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 seven numeric parameters. Our approach is different, and relies on the fact that only a restricted set of combinations of these parameters makes sense in the cinematic domain. 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.

Viewpoints are implemented in Mario as frames. 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. Thus, our formalization includes three slots used most often, 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, 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 cause an unnatural impression for the viewer, so the 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 degrees from the target's gaze vector, either to the left or to the right, respectively. By $\frac{3}{4}$ profile angle we refer to $\frac{3}{4}$ of a right angle (or 67.5 degrees), either to the left or to the right of the target gaze vector.

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.

Recall that the timeline 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 `first-vp` and `last-vp`, respectively. The formalization allows introducing additional viewpoints in the middle of intervals, but this was not utilized so far.

3.2 Cinematic Rules

For the sake of the explanation, we will use a small scene fragment from the Latin telenovela *Dulce Anna*. The rules and their formalization will be simplified; more details appear in the first author's Ph.D. thesis [12]. Assume that only the following rules in our knowledge base affect the situation:

1. If an actor is speaking, she is displayed in a frontal (30-degree) medium shot.
2. If an actor is walking, she is displayed in a long shot, from a $\frac{3}{4}$ angle.

3. Cameras don't cross the *line of interest* with a cut. In our simple example, the line of interest is the line that connects the two-dimensional positions of the two actors.
4. There are no jump cuts; specifically the angle between two consecutive shots is at least 60 degrees.[5]

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

```
(= (target (first-vp I)) a)
(= (target (last-vp I)) a)
premise: (= (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 premise; this is an example of a retractable formula.

The second rule is similar. In this case the profile-angle is a retractable premise.

Next we want to formulate the rule that states that the camera does not cross the line of interest with a cut. Note that it is impossible to detect that the line is crossed before the slot values of the viewpoints are determined. The way to handle this with Cake is by using pattern-directed invocation.

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:

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

We install a demon that is triggered whenever the pattern for the `same-side` function appears in a 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` needs 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 position is. 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 the system installs the following axiom:

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

where `side-x` is 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-degree angle difference between consecutive shots. The formalization of this rule in Cake is similar to the way we implemented the line-of-interest rule. Using pattern directed invocation, we make sure that if interval edges become cuts their viewpoints have a large enough angle difference.

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

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:

```
0 3 Mother walk-to point-1
3 6 Mother speak "So, have you eaten the
    sandwich that I have 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.

The rules require the first interval, in which Mother walks, to be displayed in long shot from a $\frac{3}{4}$ angle. In the next interval Mother speaks, so a frontal medium shot is expected. Then Mario speaks, so a frontal medium shot of Mario is required.

This, however, is impossible. For the second and third intervals, the line of interest should not be crossed, so the reasoning process makes sure that Mother and Mario would be shown from the same side of the line, one of them in front-left and the other in front-right. This results in a "jump-cut" between the first and the second intervals, i.e., a violation of the rule that requires at least a 60 degree difference between consecutive shots. The reason is that the difference between a $\frac{3}{4}$ angle and a frontal shot (30 degrees) is 37.5 degrees, which is less than 60 degrees.

Now the system needs to resolve this conflict, and it is able to do so, since some of the constraints were formulated as premises. In this example, the rules required a simple shot, i.e., that both viewpoints for the same interval will be equal. There is only way to satisfy all constraints, which is by unifying the two intervals into one shot, in which all viewpoints will display a long shot of Mother from a front-L profile angle. This is a new kind of shot, a frontal long shot, that 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 settings: the user 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 a legal solution.

Cake provides different kinds of 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 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, we would probably use a default to specify that a shot is a simple shot in a telenovela. **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 `or` clauses.

During the development of Mario, it turned out that these two mechanisms, defaults and assumptions, 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.

The algorithm that we implemented ion top of Cake, for resolving contradictions, is described in the first author's Ph.D. thesis [12], where we also show that is terminates, although it is exponential.

4 Results

We have run Mario and evaluated the results for over a hundred scenes. Most of the scenes only involved two actors, and a few included three actors. Due to a limitation 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 telenovela scene, which included 41 actions over 45 intervals, resulting in an animation sequence of 2 minutes and 20 seconds.

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 independent formulae. During the processing of the full scene, the TMS was loaded with over 10,000 terms.

Running the whole telenovela scene, which includes 41 actions and 45 intervals, takes approximately 10 seconds on a PC with 2GHz Pentium 4 processor, 1MB of RAM, running GNU Common Lisp. This acceptable for an off-line tool, which may be used interactively through iterative refinement of results.

4.1 Additional Genres

In additional to Latin telenovela, we have analyzed scenes from additional genres, mainly a more dramatic genre (based on the TV science-fiction series *The X-Files*). This introduced new challenges; mainly, it includes a much richer set of situations and locations compared to telenovela. As we have refined the knowledge base and introduced new rules, we frequently confronted the problem of contradiction between rules. It is possible to keep the problem underconstrained by using preferences and defaults rather than axioms, but the result might be an arbitrary choice made by Mario to prefer one rule over the other.

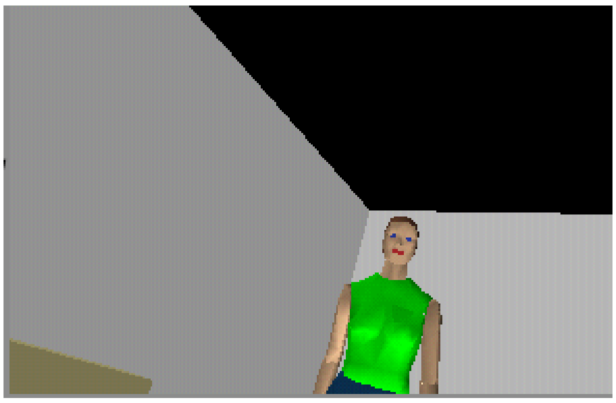

Figure 1. If a character is perceived to be threatening, a rule can specify that the character be displayed from a low angle.

Conflicting defaults is a well-known problem in non-monotonic reasoning [15], 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 into categories of high-level goals. The goals we have identified are: 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.

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. In the first, we asked Mario to prefer complex shots rather than 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 $\frac{3}{4}$ to frontal, and also preferred zooming in. The result was of a very consistent style including a lot of 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** 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.[6] The point of this camera behavior in Snow’s experimental film was to use repetitive camera rotations 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 note that we are far from a tool that would deliberately select such a style to emphasize an event, as done by Snow.

4.2 Empirical Evaluation

We have conducted an informal evaluation of Mario.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 presented five versions of the telenovela scene in 3D animation, one of which was manually prepared by our domain expert. The audience was comprised of two groups: one including graduate film students and lecturers, and the other only included subjects with no background in film.

The complete details appear in the first author’s Ph.D. thesis [12]. 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.

We also asked the viewers to rate the movies. We note that in both groups the rating for the human-made version was highest. It seems that the question whether the movie was edited by human or machine

[6] This film from 1968 was given no name, and is usually referred to as “Back and Forth.”

is misleading, whereas viewers do sense a difference, and judge that the human version is better. It is interesting to note that Liu et al. [20] also found similar results for their automated video editing system; people could not pass the "Turing test" successfully, but consistently preferred the human version.

5 Discussion and Future Work

There is a growing interest in virtual environments that display an emergence of complex phenomena. Emergent behavior is typically achieved by using evolutionary methods such as genetic algorithms. We could not use such methods in our study for two major reasons. First, they conflict with our interest in explicit knowledge formalization. Second, they require an extensive data set that is not available in our case. We have demonstrated how a form of emergent behavior, being a property of any complex system, can be achieved using a knowledge-based approach.

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. Mario does not deal with frame composition or occlusions. In the future, we would also like to deal with multiple types of scenes and locations, and with the manipulation of time. Other techniques seem in place, specifically user modelling and machine learning.

While our goal was to come up with a generic method for automated camera control, we have only looked at TV and film 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.

A major challenge is how to extend the cinematic model to interactive environments. In the scope of this research, the main effort was the knowledge formalization, and finding the best computational technique. In the case of non-linear environments, the principles have yet to be invented (discovered?). Cinematic concepts are transformed into new concepts in interactive settings; for example, in some settings camera motion transforms into navigation, and transitions are transformed into teleporting. Using our system, artists will be able to explore the cinematic principles appropriate for such genres.

We have examined a novel application of automatic camera for the automated generation of movie summaries from session in interactive virtual environments [14]; such an application poses two challenges: the first is to automatically decide what is worthwhile showing, and the second is how to show it, which is based on the research discussed here.

While camera control is one of the differentiators of film and TV from other media, it is by no means the only component of the cinematic expression. Our approach could be extended to support other components, with the goal of automated filmmaking [13].

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.

REFERENCES

[1] W. H. Bares and J. C. Lester, 'Cinematographic user models for automated realtime camera control in dynamic 3D environments', in *UM97: User Modeling: Proc. Sixth Int'l Conf.*, eds., A. Jameson, C. Paris, and C. Tasso, pp. 215–226, Sardinia, Italy, (1997).

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

[3] W. H. Bares, L. S. Zettlemoyer, D.W. Rodriguez, and J.C. Lester, 'Task-sensitive cinematography interfaces for interactive 3D learning environments', in *IUI-98: Proc. 1998 Int'l Conf. Intelligent User Interfaces*, pp. 81–88, San Francisco, California, (1998).

[4] J. Bowers, 'Crossing the line: a field study of inhabited television', *Behavior & Information Technology*, **20**(2), 127–140, (2001).

[5] A. Butz, 'Anymation with CATHI', in *IAAI-97 Proc. of Innovative Applications of Artificial Intelligence*, pp. 957–962, Providence, Rhode Island, (1997).

[6] H. Callev, *Cinematic Expression*, Optimus, 1996. In Hebrew.

[7] D. Christianson, S. Anderson, L. He, D. Salesin, D. Weld, and M. Cohen, 'Declarative camera control for automatic cinematography', in *Proc. Thirteenth National Conf. Artificial Intelligence*, pp. 148–155, Menlo Park, CA, (1996). AAAI Press.

[8] J. Doyle, 'Truth maintenance systems', *Artificial Intelligence*, **12**(3), 231–272, (1979).

[9] S. M. Drucker and D. Zeltzer, 'Intelligent camera control in a virtual environment', in *Graphics Interface*, (1994).

[10] S. M. Drucker and D. Zeltzer, 'Camdroid: A system for intelligent camera control', in *SIGGRAPH Symp. Interactive 3D Graphics*, (1995).

[11] Y. A. Feldman and C. Rich, 'Pattern-directed invocation with changing equations', *J. Automated Reasoning*, **7**, 403–433, (1991).

[12] D. Friedman, *Knowledge-Based Cinematography and Its Application to Animation*, Ph.D. thesis, Tel Aviv University, October 2003.

[13] D. Friedman and Y. Feldman, 'Knowledge-based formalization of cinematic expression and its application to animation', in *Proc. EUROGRAPHICS 2002*, pp. 163–168, Saarbrucken, Germany, (2002).

[14] D. Friedman, Y. Feldman, A. Shamir, and T. Dagan, 'Automated creation of movie summaries in interactive virtual environments', in *Proc. IEEE VR 2004*, pp. 191–198, Chicago, IL, (2004).

[15] *Readings in Non-Monotonic Reasoning*, ed., M. L. Ginsberg, Morgan Kaufmann, 1987.

[16] M. Gleicher and J. Masanz, 'Towards virtual videography', in *Proc. ACM Multimedia*, pp. 375–378, (2000).

[17] N. Halper and P. Olivier, 'CAMPLAN: A camera planning agent', in *AAAI 2000 Spring Symp. Smart Graphics*, pp. 92–100, Stanford, (2000). AAAI Press.

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

[19] P. Karp and S. Fiener, 'Automated presentation planning of animation using task decomposition with heuristic reasoning', in *Proc. of Graphics Interface '93*, pp. 118–127, Toronto, Canada, (1993). Canadian Information Processing Society.

[20] Q. Liu, Y. Rui, A. Gupta, and J. J. Cadiz, 'Automating camera management for lecture room environments', in *SIGCHI'01*, Seattle, WA, (2001).

[21] D. McAllester, 'Truth maintenance', in *Proc. Eighth National Conf. Artificial Intelligence*, pp. 1109–1116, Menlo, California, (1990). AAAI Press.

[22] M. Minsky, 'A framework for representing knowledge', in *Readings in Knowledge Representation*. Morgan Kaufmann, (1985).

[23] Y. Nam and D. Thalmann, 'CAIAS: Camera agent based on intelligent action spotting for real-time participatory animation in virtual stage', in *Proc. VSMM '99*, Dundee, Scotland, (1999).

[24] C. Rich and Y. A. Feldman, 'Seven layers of knowledge representation and reasoning in support of software development', *IEEE Trans. Software Eng*, **18**(6), 451–469, (1992).

[25] R. Thompson, *Grammar of the Edit*, Focal Press, 1993.

[26] B. Tomlinson, B. Blumberg, and D. Nain, 'Expressive autonomous cinematography for interactive virtual environments', in *Proc. Fourth Int'l Conf. Autonomous Agents*, pp. 317–324, Barcelona, Spain, (June 2000).

[27] T. Yaron, *Editing Movies*, Israeli Ministry of Culture, 1995. In Hebrew.