Dr. Richard Frost
School of Computer Science
University of Windsor
8103 Lambton Tower
E-mail: rfrost@cogeco.ca

SURVEY REPORT

Course: 60-510 Literature Review and Survey

Survey Title:
Applications of Constraint Satisfaction Problem Solving in Computer Games (Camera Control)

Instructor: Dr. Richard Frost
Supervisor: Dr. Scott Goodwin

Submitted By: Mohammed Liakat Ali
E-mail: *ali1p@uwindsor.ca*

September 12, 2007

Table of Contents

Abstract

A user observes a virtual 3D world through a virtual camera. The third person perspective camera control system provides much more information to the users than the first person perspective camera control system (Lin, Shih, and Tsai 2004; Christie et al 2005). Therefore it enhances viewer experience significantly (Lin, Shih, and Tsai 2004). A camera is controlled for particular reasons which can be represented as constraints on the camera parameters (Drucker and Zeltzer 1994). On the other hand, various kinds of constraints are used to describe relationships between the objects (Badros 1998). Constraints on camera and/or on objects of the scene in a virtual environment can define a camera shot. To find camera parameters which satisfy all the constraints, the problem can be cast as a Constraint Satisfaction Problem (CSP) (Christie et al 2005). Thus finding an appropriate shot turns into finding a solution of the CSP instance. We have conducted a literature review to find current usage of CSP solving techniques utilized in camera control systems of 3D computer games and interactive narratives. We have reviewed twenty papers. The first three background papers cover constraints and constraint solution techniques. The next three background papers cover camera control methods using various solution techniques. We have identified constraints and constraint solvers of the camera control methods in fourteen papers in the main body of the review. Third person perspective camera control system using CSP solution techniques has been implemented in all of them. Six methods achieved real-time performance, ten methods defined constraints to incorporate cinematic idioms, three methods dealt with frame coherence, four methods utilized constraint relaxation, and five methods used constraint weighting.

1. INTRODUCTION

As described in (Amerson and Kime 2000), the Hollywood cinema industry has developed a structured approach for conveying a story which can be described as film or cinematographic idioms. A film or cinematographic idiom is the template for camera positioning and motion. A film consists of multiple scenes. A scene represents a series of events occurring in continuous space and time. A scene consists of multiple shots. The duration of a shot is from the turning on to turning off of the camera. According to (Bares, Thainimit, and McDermtt 2000), a shot can also be described as a continuous stream of frames in cinematography. A frame consists of individual images.

As mentioned in (Christie et al 2005), the camera control methods used in computer games can be categorized as action replay, first person perspective, and third person perspective systems. The action replay camera system is responsible only for highlighting the important actions of the game play. A camera represents the user's viewpoint in a first person camera control system. A third person perspective camera control system can create much more sophisticated computer games than that can be achievable with the first person perspective camera control system. However, implementation of an automated third person perspective camera control system in a 3D computer game is a difficult and time consuming task.

There are usually seven camera parameters which can be controlled. They are known as seven degrees of freedom. These parameters will determine camera position, where it is directed, and its field of view (Christie et al 2005). Constraints on camera and/or on objects of the scene in a virtual environment can define a camera shot. In addition, there are ongoing efforts to incorporate cinematic effect in 3D computer games. Half of the approaches in main body of the review have utilized cinematographic idioms from the film industry and define constraints to implement cinematographic idioms for finding camera placement of a shot. To find camera parameters which satisfy all of the constraints, the problem can be cast as a CSP instance.

As described in (Bourne and Sattar 2005), a CSP is defined as a triple which consists of a set of variables, a set of domains for each of the variables, and a set of constraints. The constraints are defined as relations over the set of variables. A constraint is a subset of the Cartesian product of the variable domains. In CSP solution techniques, the problem becomes how to find an assignment or all the assignments to the variables which will satisfy all of the constraints.

In addition, the constraint solvers have to deal with issues like real-time performance, frame coherence, constraint relaxation, and constraint weighting. Moreover, visibility and obstruction avoidance are two recurring issues which arise during finding a solution to the camera placement problems. Some of the approaches covered in this survey deal with the issues with diverse techniques.

We have investigated application of CSP solving techniques to camera control system of the 3D computer games in the survey. We have also come across similar 3D virtual

application called Interactive Narratives where CSP solving techniques are equally applied to camera control problems.

In section 2, we will discuss constraints and their solving techniques as well as different camera control systems. The remaining survey report is on application of CSP to camera control problems in computer games and interactive narratives and will be referred to as the main body of the survey. In section 3, we will discuss camera control systems utilized in computer games. Section 4 will cover uses of camera control systems in development tools. In section 5, we will discuss camera control used in interactive narratives. The conclusion is in section 6. Acknowledgement, references and appendixes follow.

2. CONSTRAINTS, SOLUTION TECHNIQUES, AND CAMERA CONTROL

This section provides a glimpse on main CSP solution techniques, constraints and their solution techniques in interactive graphical applications, and a few camera control systems using various solution techniques in 3D computer games and interactive narratives. This section will be considered as background for the survey.

CSP can be used to model a wide range of practical problems. A large number of problems in AI and other Computer Science areas such as machine vision, belief maintenance, scheduling, temporal reasoning, graph problems, floor plan design, planning genetic experiments, and the satisfiability problems can be considered as special cases of CSP (Nadel 1990; Kumar 1992).

In interactive graphical applications, various kinds of constraints are used to describe relationships between objects. As a result, researchers have developed diverse constraint solving techniques (Badros 1998).

A constraint solver finds solutions using search. Variables are instantiated with values by the solver. Then it checks if the current instantiation is a consistence assignment. In case of consistent instantiation, it will continue instantiation until a solution is found or lead to an inconsistent assignment (Bourne 2006).

In section 2.1, CSP solution techniques are discussed. In section 2.2, constraints and constraint solvers used in interactive graphic applications are discussed. In section 2.3, at first camera control in interactive narratives using a software agent is discussed. Then issues related with development of camera control system in commercial computer games are discussed. Finally classification of camera control systems in 3D computer games is discussed.

2.1 On CSP Solution Techniques

The generate-and-test (GT) method is the simplest method to find a solution to a CSP. In GT, all the possible combinations of the variables are generated and tested to find a solution. The number of combinations generated and tested is equal to the Cartesian product of all the variable domains. The first combination which satisfies all of the constraints is the first solution to the CSP (Kumar 1992). All such combinations which satisfy all of the constraints are also solutions to the CSP. The cost to find the first solution or all of the solutions using GT method is enormous. There are much more efficient algorithms developed by researchers to solve CSPs.

Nadel (1990) has compared some solution algorithms empirically using two CSP instances based on the n-queens problem and what the author described as a new version of the n-queens problem called the confused n-queens problem. Thus ranking of the algorithms depends on the two selected problem instances and their processing orders. The author observes that the empirical method used in the paper may not give the same

ranking of the algorithms for an arbitrary problem instance and processing order. Hence, mathematically derived complexity analysis is required to find a general basis of comparing the algorithms.

The problem addressed in (Nadel 1990) is how to make the complexity of the solution algorithms of CSP more understandable to the readers. The author starts by referring to the preliminary version of the paper published in 1988 as 'Tree search and arc consistency in constraint satisfaction algorithms' and states that several more algorithms have been added in the current version of the paper and the presentation of the paper has also been significantly restructured. The paper has dealt with all the solution assignments. The author has conducted what he called a unified survey on the solution algorithms based on tree search, arc consistency, and hybrids of the two methods. An arc consistency algorithm makes a directed arc (x, y) of a constraint graph consistent by removing all the values from the domain of the variable x for which there is no corresponding value in the domain of the variable y to satisfy the constraint C(x, y). Both parameterized full and partial arc consistency algorithms are introduced. Parameterized arc consistency procedures can be combined with tree search algorithms. Hence arc consistency processing can be done on the sub problems rooted at each individual search tree node. The author observes that implication of non parameterized arc consistency algorithms are either the algorithms are adequate to solve a CSP instance or can only be used for preprocessing before application of a search algorithm to solve a CSP instance.

The paper is a survey of solution algorithms of CSP. Tree search solution algorithms Backtracking, Backjumping, and Backmarking are discussed. The author observes that several important algorithms such as classic backtracking, and Haralick's full and partial Lookahead algorithms when rearranged take hybrid form. But all the arc consistencies of rearranged algorithms do not achieve full arc consistency at the tree nodes. In the spirit of full arc consistency procedures such as AC1, AC2, and AC3, new partial AC procedures have been called AC ⅕ , AC¼, AC⅓, and AC½ based on the degree of partial arc consistency achieved by the algorithms. All of the arc consistency algorithms known at that time are discussed. Then nine hybrid algorithms with full arc consistency TSRACi, TSACi, and RFLi (i = 1, 2, 3) and four hybrid algorithms with partial arc consistency FL, PL, FC, and BT are discussed. The n-queens and confused n-queens problems were formulated as complete, binary CSP instances to compare the complexity of fifteen tree search and hybrid algorithms empirically for all the solution assignments. A tree searching algorithm called Backmarking by Gasching and a hybrid algorithm called Forward Checking by Haralick are reported as most efficient ones. Nadel observes that the arc consistency processing at the nodes for hybrid algorithms increases number of nodes. Hence the methods using arc consistency should sufficiently reduce average work per node so that less work over the whole tree is needed. The author claims that little arc consistency by hybrid algorithms reduce workload over whole tree and unified view taken for the survey suggests several new algorithms. One of the new algorithms is considered as the best one based on preliminary test results. The new algorithm has been named tentatively as 'TSSTAC3' for 'Tree Search + Singleton Target Node AC3'.

The author identifies the following future work:

- To find new applications of CSP.
- To understand theoretically and empirically existing solution algorithms of CSP.
- To develop new solution algorithms of CSP.
- To generalize the problem and its algorithms.

Kumar (1992) has observed that a CSP can be solved by classical backtracking algorithm in any circumstances. But the algorithm does not learn from failures in different nodes. As a result it has very poor performance. The author has identified reasons of poor performance and presented various techniques to improve performance of the algorithm. Hybrids algorithms consist of backtracking with varying degree of arc consistency have also been investigated.

The problem addressed in (Kumar 1992) is how to make solution techniques of CSP more accessible to the readers. The paper has dealt with first assignment to the variables which satisfy all of the constraints. The scope of discussion of the paper is further limited to the solution techniques for a class of CSP called binary CSP where each constraint is either unary or binary. A unary constraint involves a single variable and a binary constraint involves two variables. The author provides a brief overview in tutorial format of many solution techniques utilized to solve the problems in AI and other Computer Science areas cast as a special case of the CSP.

The paper is a survey of solution techniques of CSP. At first, the generate-and-test paradigm is discussed. Then more efficient backtracking algorithms are discussed. But the algorithms have also exponential complexity. Thrashing and redundant work performed are identified as two major drawbacks of the standard backtracking algorithm. Node inconsistency and lack of arc consistency are described as two main causes of thrashing. Node inconsistency arises whenever the domain of a variable contains a value that does not satisfy a unary constraint on the variable. The author suggests that node inconsistency can be eliminated by removing all the values from the domains of each variable that do not satisfy a unary predicate and arc consistency can be achieved by making each arc of constraint graph consistent. The following three techniques are identified to avoid redundant work and improve performance of the backtracking algorithm:

1. Constraint propagation at search node.
2. Reason maintenance or intelligent backtracking.
3. Ordering of variable and instantiation of different values.

Using the constraint propagation technique, a given CSP can be transformed into another CSP which has a smaller search space compare to the original one in such a way that the same failures are not encountered again. After describing AC1, AC3, and k-consistency, the author investigates the conditions for the existence of search order that is backtracking free. Then combination of constraint propagation and backtracking techniques is discussed. The author states that only a limited form of constraint propagation with backtracking can be applied for better results. Next, intelligent backtracking and truth maintenance techniques are discussed. Using these techniques,

information about a failure is kept updated and used during search of the remaining search space. Truth maintenance methods deal with developing general techniques to attach justifications or assumptions to the inference. At last, approaches using variable and value instantiation ordering techniques are discussed. The author observes that application of variable ordering techniques will move failures to upper levels of the search tree. On the other hand, application of value instantiation ordering techniques will make a solution of CSP moved to the left of the search tree. The author claims that thrashing of the backtracking algorithm can be totally eliminated with much more cost than that incurred by the simple backtracking if any of constraint propagation, reason maintenance or intelligent backtracking, or variable ordering or value instantiation ordering techniques is applied to an extreme. On the other hand, a simplified version of the techniques can be used together with backtracking to reduce overall search space efficiently. But the optimal combination of these techniques will be problem dependent.

The author identifies future work as to find an optimal combination of various improvement techniques of backtracking algorithm for different problems.

2.2 On Constraints and Solving Techniques

Badros (1998) has dealt with classes of constraint used in interactive graphical applications, their solution techniques, and classification of the constraint solvers. The author identifies following common issues the constraint solvers must address:

- Under constrained systems must be handled.
- Spatial stability must be maintained.
- Sharing data structures to provide the equality or coincidence-of-points constraint.
- Incorporating variable aliasing optimization.
- Splitting of interactive constraint solvers into planning or compilation, and execution stages.

Badros (1998) has observed that none of the interactive graphical applications discussed in the paper allows disjunctive constraints. But support for disjunction can be added. The author suggests the following ways to make backtracking algorithms able to handle disjunctions:

- Chronological backtracking algorithm is to be implemented as an incremental one.
- Using minimum remaining value optimizations.
- Using a conflict set of an added constraint.
- Preserving data structures from prior solutions.
- Maintaining constraints in a disjunction in the tableau, but manipulating only their weights.
- Exploiting domain specific knowledge.
- Performing additional preprocessing of the stable constraint set.

The problem addressed in (Badros 1998) is how to make readers familiar with the classes of constraints and solution techniques used in the interactive graphic applications such as drawing, graph layout, visualization, and animation systems. The author summarized

performance of backtracking algorithms and suggested several ways to improve their performance in the interactive graphical applications.

The constraints relevant to the geometric applications are identified as: 1) “linear equalities”, 2) “linear inequalities”, 3) “linear geometric”, 4) “geometric”, 5) “geometric (on camera image)”, 6) “geometric in complex plane”, 7) “point-on-object”, 8) “coincident”, 9) “arbitrary acyclic”, 10) “Horizontal relationship between pairs of points (HOR)”, 11) “Vertical relationship between pairs of points (VER)”, 12) “Parallel relationship between pairs of line segments (PARA)”, 13) “Congruence relationship between pairs of line segments (CONG)”, and 14) “Visual Organization Features (VOFs)”.

The constraint solving techniques used are identified as: 1) “relaxation”, 2) “numeric”, 3) “iterative numeric”, 4) “optimized iterative numeric”, 5) “direct numeric (QOCA)”, 6) “differential methods”, 7) “symbolic”, 8) “spring simulation”, 9) “graph-layout, direct & iterative”, 10) “extreme-bound propagation”, 11) “Degree of Freedom (DOF analysis)”, 12) “Constraint Logic Programming with Real arithmetic constraints (CLP(R)-like)”, 13) “Local Propagation (LP)”, and 14) “LP w/o planning”.

The paper is a survey of the classes of constraints and their solution techniques used in interactive graphical applications. The constraints and their solvers used in various systems are shown in table 1. The author classified the constraint solving methods and compared their expressiveness and performance. A visual classification of interactive constraint solvers is given in figure 1 where solvers and sub-solvers are grouped using different kinds of lines, relationships between solvers are shown with the help of different kinds of arrows, and proximity in the figure roughly indicates relatedness. The author observes that the constraints are used to keep relationship among on-screen objects and using constraints is a declarative means for specifying relationship that designers or users wish to hold true, the declarative specification of desired relationship is the fundamental strength of using constraints, and the backtracking algorithm is not successfully used to find solutions in the interactive graphical applications. The author claims that expansion of the classes of the constraints used in an application is a big challenge because the classes of constraints used by the application are restricted by its constraint solver.

The author identifies the following future work:

- To develop fast, expressive and understandable reusable constraint solvers.
- To reuse constraint solvers.
- To make debugging of constraints easier for the users.

	System	Author (Year)	Constraints supported	Solving technique	Performance
Drawing	Sketchpad	Sutherland (1963)	geometric	LP, relaxation	$O(n)$, $O(n^2)$
	IDEAL	Van Wyk (1982)	geometric in complex plane	LP w/o planning	$O(n^2)$
	Juno	Nelson (1985)	CONG, PARA, HOR, VER	iterative numeric	$O(n^3)$
	Juno-2	Heydon and Nelson (1994)	CONG, PARA, HOR, VER	optimized iter. num.	$O(n^3)$
	Briar	Gleicher and Witkin (1994)	points-on-object,coincident	differential methods	$O(n^3)$
	Unidraw	Helm et al. (1995)	linear (in)equalities	direct numeric (QOCA)	$O(n^3)$, $O(n^2)$
	GCE	Kramer (1992)	geometric	DOF analysis	$O(ng)$, $O(n \log n)$
	Chimera	Kurlander (1991)	geometric	symbolic, numeric	$O(n^2)$
	Pegasus	Igarashi et al. (1997)	geometric	CLP(R)-like	$O(2^n)$
Graph	GLIDE	Ryall et al. (1997)	VOFs	spring simulation	polynomial
	CGL	He et al. (1996)	linear (in)equalities	iter. numeric	> 1 sec (tree $n = 16$)
Visualization	TRIP*N*, IMAGE	Takahashi et al. (1998)	linear geometric	graph-layout, direct & iterative	"needs to be faster"
	ICOLA	Oster & Kusalik (1998)	linear inequalities	extreme-bound propagation	$O(n + v)$
	Penguins	Chok & Marriott (1998)	linear (in)equalities	direct numeric (QOCA)	interactive ($n \leq 700$)
Animation	TLCC	Gleicher & Witkin (1992)	geometric (on camera image)	differential methods	$O(n^3)$, non-interactive
	Animus	Duisberg (1987)	arbitrary acyclic	LP, relaxation	$O(n)$, $O(n^2)$
	JIM, Parcon	Griebel et al. (1996)	linear (in)equalities, geometric	iterative numeric	< 1 sec ($n \leq 100$)

Table 1: List of constraints and solvers in interactive graphical applications (Badros 1998; page 3; table 1)

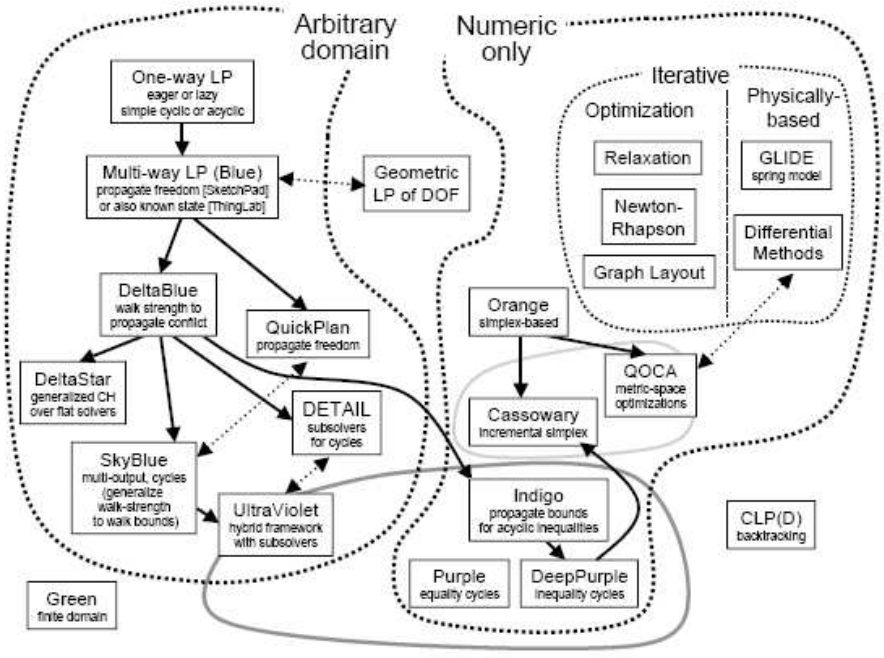

Figure 1: Classification of Interactive Constraint Solvers (Badros 1998; page 10; figure 1)

2.3 On Camera Control Systems

The authors in (Hornung, Lakemeyer, and Trogemann 2003) have dealt with visual emphasis of narrative content and application of dramaturgical concepts of cameras and classical cinematography idioms to interactive storytelling. As described in the paper,

human perception and interpretation of a scene is strongly influenced by the visual presentation of a narrative. In virtual interactive storytelling, rigid geometrically oriented techniques are not able to emphasize narrative contents. The authors have tried to derive a basic formalization of narrative expressiveness of camera and cinematographic rules. They have implemented the formalization in a camera agent. The camera agent uses a neural network called a perceptron to choose appropriate camera shots for a given situation within the current narrative context.

The problem addressed in (Hornung, Lakemeyer, and Trogemann 2003) is how to transfer cinematographic knowledge of camera control to interactive narratives and games. The authors formalize the dramaturgical means of expression of movie camera systems to be applied to interactive narratives. They identify a set of basic dramaturgical principles and use eight parameters of events to communicate information about states of a story between a narrative application and a camera agent. A story is represented within the camera agent as narrative events in such a way that a narrative event corresponds to a single story event. The narrative application will send information regarding its states as narrative events to the camera agent. After receiving an event, the camera agent re-computes its internal representation of the narrative based on the events received so far. Then the camera agent selects an event as the active one for visualization based on the history of the narrative, coherence between the events, event priority etc. Next, the decision module of the camera agent selects matching shots from user-defined shot library to be added to a priority list of the active event. Finally, the action-realization module of the camera agent computes the camera position, orientation etc. using a cinematic rule for the shot with highest priority for the active event and transfers the information back to the narrative application for visualization. If a shot can not be realized, the next matching shot will be considered.

The authors developed a real time autonomous camera agent. To test the camera agent, the authors modified an existing computer game called Half-Life. Modified Half-Life will generate a narrative event whenever a character changes its internal state. The authors claim to have conducted an experimental evaluation by letting professional cinematographer and non professional audiences view both the original first-person view of the game and the shots created by the camera agent. The result of the experiment is reported as “very convincing and significantly enhanced the narrative experience of Half-Life for the spectator”. The authors claim that their system, as a part of an interactive narrative environment, demonstrated the practicality of the system.

The authors identify the following future work:

- To understand and formalize more complex cinematic concepts.
- To integrate planning techniques to allow sophisticated reasoning about visual outcome of camera shots such as geometric constraints within a scene to support the narrative.

Giors (2004) has dealt with implementation issues of camera control systems in commercial computer games. The author observes that the camera system is often neglected during development of commercial games but Full Spectrum Warrior (FSW)

has one of the most advance camera control system among commercial computer games. The author makes the following recommendations to implement effective camera control system in a commercial computer games:

- Usability should prevail in any conflict between aesthetics and usability except during cut scenes.
- All motions should be smooth.
- Always avoid quick camera panning except under user control because it has disorientation effect on the user.
- A layered design with multiple cameras is to be tried.
- Camera controls by the program and the user should be kept separated.
- User controls of camera should have as much freedom of motion as possible.
- Visual feedback, tuning, and debugging tools are to be given.
- Alternate solution to camera problems even by means outside of the camera system should to be considered.

The problem addressed in (Giors 2004) is how to implement an effective camera control system in commercial computer games. The author observes that the basic Proportional Controller (PC) is a common camera motion control system in commercial computer games. PC has nice arrival characteristics. The disadvantages of PC are abrupt exit characteristics and lag of current camera position behind the moving targets. To solve the problems of PC, Modified Proportional Controller (MPC) is used to control camera motion in FSW game. MPC controls exit characteristics by limiting acceleration and lag is corrected considering velocity of the target points.

The author describes the camera system of FSW. FSW has several cameras. Each camera is derived from an abstract base class. The cameras managed by a camera manager may operate simultaneously or independently. The main camera represents the primary user controlled view and the fly-by system. The other cameras are for cinematic, in-game events, and playback. The cinematic camera can move between preset positions in addition to the movements of normal in-game view system. The camera system remains under control of the user all the time except at the beginning of a fly-by. FSW has an auto-look feature to look around corners. This feature will handle occlusion using ray-casting from the camera position. On the other hand, FSW uses ray-casting from the 'lookat' object to avoid collision during normal view and fly-by action. FSW uses camera cuts to avoid passing through obstacles and whenever the destination is beyond a preset distance. The author observes that multiple independent cameras are more useful due to different viewing requirements and one general purpose camera that does everything is likely to lead to a highly complex system. The FSW camera system is affected by over 100 parameters which pose a tuning problem for the designers or programmers. The author claims that unique solutions to several camera control problems were found for FSW due to special attention being given to the issues related to the implementation of camera system during development of the game.

The author identifies the following future work:

- To re-implement the main camera with several cameras to simplify tuning problem.

- To integrate traditional cinematic camera ideas from the film industry.
- To replace linear ray-hits by volumetric examination to improve the operations of collision avoidance, auto-look etc.
- To update feedback system during tuning process.

The authors in (Christie et al 2005) have dealt with the virtual camera planning. A virtual world is viewed through a virtual camera. To view the virtual environment, the camera parameters are to be specified. Finding values of the parameters is the task of camera planning. The authors call for particular attention to camera positioning and movements to convey the desired information. They observe that constraint and optimization based solution techniques can add easily new constraints or objective function for any specific property. But computation cost for the techniques are much higher. For these approaches, how to assign the correct weight for each property and how to aggregate the weights in the objective function remain two important issues. The authors have considered the real-time camera planning analogous to the documentary cinematography. For both the cases behaviour of scene elements should be presented without modification to the positions or orientations of the elements.

The problem addressed in (Christie et al 2005) is how to make automated virtual camera planning techniques more understandable to the readers. The authors have presented in the paper an overview of automated camera planning techniques and structured the survey on the expressivity of the set of properties and the characteristics of the solving mechanism. Three levels of image properties are proposed. They are geometric, perceptual, and aesthetic. However most of the actual systems rely solely on the geometric properties.

The paper is a survey of camera control systems. At the beginning, the requirements of camera planning systems based on shot properties are discussed. The camera control systems are classified as first person, third person, and action replay camera systems. Then the camera planning techniques around expressivity and solution mechanisms are discussed. The techniques based on on-camera and on-screen properties of expressivity are described. The on-camera methods include both static and dynamic systems. Four types of existing camera planning approaches based on solving mechanisms are described as algebraic, interactive, reactive real time, and optimization and constraint-based systems. The optimization and constraint-based systems are classified as complete, incomplete, and hybrid approaches. The complete method includes hard constraint CSP and hierarchical constraint methods. The incomplete method includes constraint-optimization and pure optimization methods. The authors observe that solution technique adopted by the approaches constrains both geometric abstraction and expressiveness. The authors claim that they have identified and described in the paper the principle shortcomings of the existing techniques of automated camera planning.

The authors identify the following future work:

- To improve quality of abstraction of 3D objects with an adequate model as well as an effective solving mechanism.

- To develop a model for constraint based systems to decide weight of each value and aggregation of the weights in a single objective function.
- To develop a cognitive model which will incorporate perceptual and aesthetic properties to assist users to construct time and space characteristics of the virtual world in their mind.

This section has dealt with comparison of different CSP solution techniques and covered the constraints used along with their solution techniques in the interactive graphical applications. This section has also covered a camera control system using a camera agent in interactive narratives, camera control systems in commercial computer games, and a survey on camera control systems. Comparison of the methods covered in this section is given in table 2.

Paper	Description	Comments
(Nadel 1990)	A unified survey of tree search, arc consistency, and hybrid methods of the CSP solution techniques; Deals with all solutions of CSP	Introduced parameterization of arc consistency procedures; Compared complexity empirically of 15 solution algorithms to solve the n-queen and the confused n-queen problems cast as CSP instances
(Kumar 1992)	A survey of the solution techniques of binary CSP; Discussed intelligent backtracking, truth maintenance, variable ordering, and order of value instantiation for efficient search; Deals with first solution of CSP	Identified major drawbacks of the standard backtracking algorithm; Investigated optimal use of arc consistency procedure and search; Presented in tutorial format
(Badros 1998)	A survey of constraints and their solution techniques used in interactive graphical applications; Classified the solvers used	Identified the constraints used; Identified common issues the constraint solvers must address; Suggests ways to make backtracking algorithms able to handle disjunctions
(Hornung, Lakemeyer, and Trogemann 2003)	Derived a basic formalization of narrative expressiveness of camera and cinematographic rules to be applied into interactive narratives; Used eight parameters of events to represent states of a story	Identified a set of basic dramaturgical principles; Implemented a real-time camera control agent
(Giors 2004)	Deals with issues encounter during camera control system implementation of commercial computer games	Made several recommendations to implement effective camera control system; Observed that multiple independent cameras are more useful
(Christie et al 2005)	A survey of the virtual camera planning; Deals with expressively and solution mechanisms	Proposed three level of image properties; Identified four types of camera planning; Observed that solution techniques constrains both geometric abstraction and expressiveness

Table 2: Comparison of the methods

3. CAMERA CONTROL IN COMPUTER GAMES

This section covers five methods and provides a brief overview of the application of camera control systems in 3D computer games using CSP solution techniques. In addition, ideas of constraint relaxation, frame coherence, and constraint weighting to find solutions of the camera control problems used by the methods are covered. Application of cinematic idioms in 3D computer games using constraints and real-time performance of the methods are also covered.

One method (Drucker and Zeltzer 1994) is implemented in a static environment and discussed in section 3.1. The remaining methods are implemented in dynamic environments. The dynamic methods that incorporate constraint relaxation, frame coherence, and constraint weighting are discussed in section 3.2, 3.3, and 3.4 respectively.

3.1. Static Game Environments

The authors in (Drucker and Zeltzer 1994) have dealt with behaviours of virtual camera which are described in terms of task level goals and constraints. One of the first steps in visual interface design is identified as task analysis of the application. The authors suggest that the following generic visual operations should be supported by all visual interfaces:

- Orientation for various coordinate frames.
- Navigation through the scene.
- Exploration of unknown 3D spaces.
- Presentation to the observers.

The Virtual Museum described in (Drucker and Zeltzer 1994) has been implemented in a static virtual environment. A user of the Virtual Museum can select rooms of the museum she wants to visit. Then rooms to be visited during a tour are located along a path. For each room of the path, the paintings and other artifacts to be visited are ordered based on entry and exit doors of the room. The camera moves along the path to visit all the selected rooms and display the paintings and artifacts.

The constraint solution technique of (Drucker and Zeltzer 1994) is an incomplete search method and cannot handle constraint failures. The solver uses a camera optimizer based on the Feasible Sequential Quadratic Programming coded in C (CFSQP) (Christie et al 2005).

The problem addressed in (Drucker and Zeltzer 1994) is how to implement intelligent camera controls in a 3D virtual environment. The authors start by referring to their previous work in (Drucker, Galyean, and Zeltzer 1992) on procedural framework for camera control and state that problems in devising a general procedure for camera control led them to formalize the constraint-based framework. They also state that the paper is based on camera specifications used in the computer graphics and path finding methods used in the robotics. The authors present a design methodology for a framework to

control the camera in a 3D virtual environment. The authors observe that a camera is controlled for particular reasons which can be represented as constraints on the camera parameters. The constraints can be identified from analysis of task requirements. Thus at first, task requirements for a specific environment are to be analyzed. Next, camera modules are built with constraints identified from the task requirements. The concept of camera module of the paper is analogous to the concept of shot in the cinematography. Finally, a group of user interfaces are to be connected to the framework. The following are examples of constraints used in the method:

1. "maintain the camera's up vector to align with world up".
2. "maintain height relative to the floor".
3. "maintain the camera's gaze (i.e., view normal) toward a specified object".
4. "maintain the camera's position on a collision free path through world".

The authors implemented a camera control framework for a virtual museum. The Virtual Museum system consists of a general interpreter, a built-in renderer, an object database, and a network of camera modules along with user interfaces. The authors identified tasks to be performed in the virtual museum. They describe the camera modules which will encapsulate the tasks as constraints. A camera module consists of an initializer, a controller, a list of constraints, and state components. A constraint solver finds final camera parameters for a camera module satisfying all the constraints of the module. Active camera module handles all of the user inputs. The authors describe the path planning camera module to navigate through the museum. It is identified as the most complicated module. Path planning is done in steps. The A* algorithm is used to find which rooms are to be visited. The path through a room is found using a numerical navigation function. Tour plans though the virtual museum is also described. The interpreter and user interface were implemented in TCL and TK respectively. The test environment was a W3D system. The W3D system is an extension to the 3D virtual environment software testbed of MIT. The authors claim that using their method special constraints can be designed and combined with other constraints for camera control.

The authors identify the following future work:

- To add more efficient path planning algorithm to deal with totally dynamic environment.
- To add variety of constraints from cinematography into the camera framework.

The results of the paper have been referred to by (Lin, Shih, and Tsai 2004).

3.2. Solvers with Constraint Relaxation

The authors in (Bares and Lester 1999) have dealt with problems of the visualization interfaces. Interactive 3D worlds appear in many entertainment, educational, and training systems. Real-time visualization interfaces need to respond to user-specified viewing requests in complex environments. An intelligent 3D visualization interface uses an automatic camera planner to position the camera to view the scene. The authors observe that an automatic camera planning system should satisfy the following four requirements:

1. "User specific viewing goals".

2. “Real-time behavior unpredictability”.
3. “Environmental complexity”.
4. “World non interference”.

An automated camera planning based on constraint satisfaction approach casts viewing goals as constraint satisfaction problems. The visualization interface developed in (Bares and Lester 1999) employs a partial constraint framework. The users can select the objects with constraints on them as the viewing goal.

The constraint solver supports relaxing weak constraints, multiple shots, and composite shot with insets. In (Bares and Lester 1999) the hierarchical constraint solving strategy is used (Christie et al 2005).

The problem addressed in (Bares and Lester 1999) is how to enhance abilities of the visualization interface of 3D virtual environment to create real-time dynamic visualizations that effectively respond to viewing request of the users. The authors propose a real-time camera visualization interface system for dynamic 3D world called CONSTRAINTCAM. The proposed intelligent visualization interface uses a partial constraint based framework. The main modules of the system are the Constraint Analyzer, Constraint Solver, and Multi-Shot Frame Composer. Visualization goal of a user is expressed as a constraint problem. Viewers can select which subject(s) to be viewed, specify constraints for each subject(s), select cinematic style or pace etc. Given the viewing goal, the Constraint analyzer will determine a consistent region where all the constraints will be satisfied. The Constraint Solver will search the consistent region to find a camera placement that will satisfy all the constraints. If such a solution is not possible, the Constraint Solver relaxes pairs of weakest incompatible constraints in steps to find a solution. If a solution is not found even after constraint relaxation, the problem is decomposed to find multi-shot solution. In that case, the Multi-Shot Frame Composer creates a sequential or composite visualization solution. The method supports the following four constraints on the subject(s) to be viewed:

1. “Vantage Angle”.
2. “Viewing Distance”.
3. “Occlusion Avoidance”.
4. “Room Enclosure”.

The authors describe the system with an example interaction in an interactive 3D testbed. The example interaction consists of multiple autonomous characters interacting in a dense cityscape. The test environment was a Pentium II 333MHz processor with Permidia2 OpenGL accelerator. The authors report that the frame rate achieved by the method was approximately 7 to 15 frames per second. The real-time performance of the method and the results of an informal focus group study with viewers are reported as encouraging. The authors claim that by using constraints relaxation and creating multi-shot solutions, the system can compute near optimal solution to difficult problems.

The authors identify the following future work:

- To add a direct manipulating interface to select subject(s) and values of any of the constraints.
- To use the silhouettes instead of bounding boxes of irregular and concave polyhedral occluding objects to handle occlusion avoidance.
- To develop a more complete representation of distance of potential occluding objects.

The results of the paper have been referred to by (Bares, Thainimit, and McDermott 2000).

The concept of constraint relaxation has also been supported in the camera control systems described in (Amerson and Kime 2000), (Halper, Helbing, and Strothotte 2001), and (Lin, Shih, and Tsai 2004) which are fully discussed in sections 5, 3.3, and 3.3 respectively.

3.3. Solvers with Frame Coherence

The authors observe in (Halper, Helbing, and Strothotte 2001) that little work has been done to immerse the user in an environment by controlling an object and keeping the camera control system invisible to the user at the same time. The system should communicate important visual goals so that user will never run into an unknown situation. They also observe that all shots in interactive applications are presented to the user immediately without editing. Hence to plan shots and transitions, the director module should know what is probably going to happen next. Thus a predictive camera planning system is used in the paper. If a good camera position cannot be found in the scene evolved up to that moment due to a geometric constraint, a less satisfactory camera position has to be used. Frame coherence is given due consideration in the paper. The method will choose a partially-satisfied camera position that is closer to the current camera position to have smooth animation.

In (Halper, Helbing, and Strothotte 2001) a dynamic approach is used to define constraints. Constraints can be chosen in various combinations. An ad-hoc incremental solver is used for constraint solution (Christie et al 2005).

The problem addressed in (Halper, Helbing, and Strothotte 2001) is how to implement an automatic and dynamic camera control system in computer games which obtains a balance between optimality of camera setting and smooth transition with appropriate cuts. The camera module of the method consists of a Director and a Predictive Camera Planner. Based on past trajectory and acceleration, scene and object states at a future time are calculated by the Planner. The current trajectory is modified by the Planner so that the camera will be at the calculated position at that time. It achieves frame coherence by computing a new camera state using existing camera states. The camera position for the next frame is estimated along the path. Then it requests the Director for a setting for the future time. The events created by the action module of the game pipeline are used by the Director to select templates from its Shot Library with the help of its Template Selector.

To fit successive shots together, various transition rules are utilized to select templates. The Director selects constraints using the templates from its Shot Template for the Constraint Solver of the Planner and uses its Emotion Template to influence the result of camera shots. Then the Constraint Solver is applied by the Planner on the estimated position to find the camera state for the next frame. The Constraint Solver finds a solution for the constraints in hierarchal order in such a way that a successive constraint solution influences previous solutions minimally. Each constraint has an optimal setting, a tolerance region, and a relaxation parameter. The relaxation parameter will determine how far the camera can be placed from the optimal value. The camera is placed at the boarder of the tolerance region of a constraint if it lies outside of the region. The occlusion avoidance is handled automatically by generating a potentially visible region using coloured polygons and an image buffer. The brightness is used to indicate the preferred region to avoid occlusion. The system supports the following seven constraints:

1. "Height angle".
2. "Level at".
3. "Facing".
4. "Angle to line of interest".
5. "Size".
6. "Visibility".
7. "Look at".

The authors set up experiments to explore virtual environments. An exploration template was implemented using four constraints namely 'level at', 'size', 'visibility', and 'look at'. With this constraint set, the camera should follow the character at its height and keep it in view without obstruction all the time. Three scenarios namely a helicopter flying through a city, a human exploring inside of a medieval building, and a bee flying through a highly cluttered attic were explored using the template. The authors report that the camera performed remarkably well in all the cases as it avoided obstructions and collisions, and was able to adjust appropriately. They claim that their camera system is the first frame-coherent constraint-based camera engine and produces "nearest-best-fit" frame coherent camera solution in real time by considering future conditions.

The authors identify the following future work:

- To solve for various visual goals based on the coherent results produced by the method.
- To develop new types of camera management by adding and learning new techniques using evolutionary design to achieve more sophisticated navigational goals and add more predictive techniques.

The results of the paper have been referred to by (Lin, Shih, and Tsai 2004).

The authors in (Lin, Shih, and Tsai, 2004) have presented another constraint-based event-driven camera approach which combines camera control in real-time 3D computer game with cinematic principles. The camera movements will be automatic based on the cinematographic principles without any user manipulation. The authors utilize a sequence of shots as per cinematic heuristics to capture a scene. The cinematic heuristics

considered in the paper are ‘do not cross the line’, ‘avoid jump cut’, ‘use establishing shots’, ‘let the actor lead’, and ‘break movement’. The camera modules are included to handle shots in ‘fixed’, ‘track’, ‘pan’, ‘point of view’, ‘over the shoulder’, ‘close shot’, ‘gun fight’, and ‘close fight’ situations. The focus of the paper is to improve game experience in a playable game by using cinematic camera control.

The problem addressed in (Lin, Shih, and Tsai 2004) is how to apply basic cinematic camera control in 3D computer games to enhance the viewer’s gaming experience. The authors begin by referring to (Drucker 1994), (Drucker and Zelter 1995), (He, Cohen and Salesin 1996), and (Halper, Helbing, and Strothotte 2001) and state that the camera control concepts of the above mentioned papers are used in this paper. They also refer to (Drucker 1994), (Drucker and Zelter 1995), (Bares, Thainmit, and McDermott 2000), and (Halper, Helbing, and Strothotte 2001) and state that the solver concepts of the above-mentioned papers are used in this paper. The proposed event-driven computer game system with cinematic camera control consists of a real-time application, a camera control system, and a renderer. The camera control system consists of camera modules and descriptions of shots. A camera module represents a cinematographic shot. It solves the constraints defined by the shot. It handles camera placement and transition based on user input, its constraint list, and parameters from the application. The system has eight camera modules to handle any situation. A description of a shot is a sequence of shots. It represents a cinematographic scene and is a finite state machine of camera modules. The descriptions of shots are organized hierarchically. At each time tick, the game application will send the camera control system a list of parameters which are relevant for the time tick. The camera parameters will be handled by the active camera module. By using the parameters and the current state of camera setting, the camera control system produces camera specification for the time tick and sends the specification to the renderer. At the same time tick, the game application supplies the renderer with content of the game which includes static geometry, materials, light, and the story. The renderer uses the information for the same time tick from both the game application and the camera control system to display.

The authors describe some cinematographic principles and explain various descriptions of shots using their camera modules. Frame coherence and obstruction avoidance systems are described. If frame coherence is required, camera position and angle are interpolated. For obstruction avoidance, a line from camera to the bounding sphere of the actor is used to determine if there is an obstruction or not. In addition, if smooth camera movement is required, the line intersection test is replaced with a cylinder to detect an obstruction before it happens. Constraints and constraint solver used in the paper are same as those used in (Halper, Helbing, and Strothotte 2001). The system was implemented in C++ and based on Half-life game engine. The test environment was a Pentium 4, 2133MHz processor with 256MB RAM PC. The authors explained improvement of few captured shots using the camera modules. The authors claim that the camera control system can generate shots automatically and arrange the shots to create cinematic effect which is suitable for 3D computer game.

The authors identify the following future work:

- To generalize the camera system to fit in to any graphic engine in place of current Half-Life game engine.
- To integrate more techniques like emotional state to extend the ability of the system.
- To consider the use of expert system to deal with complex camera planning due to too many conditions.

The concept of frame coherence has also been supported in the camera control system described in (Bourne and Sattar 2005) which is fully discussed in section 3.4.

3.4. Solvers with Constraint Weighting

The authors in (Bourne and Sattar 2005) have described an approach where constraint weighting is applied to solve autonomous camera control problems of computer games. They observe that maintaining visual coherence should be the main criterion of an effective camera control system. The following four requirements of a successful automatic camera control system are identified in the paper:

1. "Autonomy".
2. "Reactive".
3. "Real-time".
4. "Dynamic".

The authors observe in (Bourne and Sattar 2005) that proposed camera control methods in the literature use ad-hoc combinations of various camera control methods with different visibility maintenance systems. Integrating constraints-weighted local search and ray-casting for visibility can be an effective unified methodology of camera control. The authors have used the methodology with a minimal constraint set for three variables constraint satisfaction problem and weighted average targeting system to satisfy all of the above mentioned four requirements.

The problem addressed in (Bourne and Sattar 2005) is how to implement autonomous camera control for third-person perspective computer games with manageable implementation complexity and system usability. The authors begin by referring to their previous approach of (Bourne and Sattar 2004) and state that this paper extends their previous work. The authors applied constraint weighting techniques in the paper. Each of the constraints has two parameters. One is for the desired value and another is for the weight of the constraint. The weights will indicate preferences among infeasible constraints. The weights are unique and normalized. Different kind of camera behaviours can be achieved by manipulating the weights. The cost of a constraint is the weighted difference between its desired value and current potential solution value. The total cost of a solution for a camera position in a frame is the sum of the costs for individual constraints except 'frame coherence' constraint. An incomplete Stochastic Local Search technique is applied to find the lowest cost solution for the constraints for a frame. The search method makes a pre-defined numbers of moves and returns the best solution found so far as optimal solution. The 'frame coherence' constraint prevents camera movement

if cost improvement does not exceed the constraint's value and act like a simple acceleration and deceleration mechanism. A meta-level visibility constraint using ray casing from the target to the camera is applied to influence the underlying constraint solver for occlusion avoidance. The proposed system supports the following four constraints:

1. "Height".
2. "Distance".
3. "Orientation".
4. "Frame Coherence".

The camera system was implemented within a 3D game engine with dynamic environment support with changing or multiple target objects. The test environment was an AMD Athlon 2800+ processor, 512MB RAM, Windows XP Service Pack 2 PC. The test 3D world has around 225,000 triangles and 350 bounding volumes where a combination of scripted and interactive control of the target object is used. The authors state to have conducted an experiment and report that the camera system takes 0.00094 seconds per frame on average to perform both the visibility test using ray-casting and constraint solving. The authors claim to have demonstrated that autonomous camera control can be significantly simplified, and the gap between implementation and usability requirements can also be significantly reduced by their technique.

The authors identify the following future work:

- To develop a more efficient heuristics for local search algorithm.
- To extend the viewing properties of the camera with new constraints.
- To generate constraint weights by using a genetic algorithm so that the interactive camera can match a scripted camera movement.
- To investigate application of the camera system into more sophisticated autonomous cinematography system.

The concept of constraint weighting has also been supported in the camera control systems described in (Bares and Lester 1999), (Amerson and Kime 2000), (Bares et al 2000), and (Bares, Thainimit, and McDermott 2000) which are fully discussed in sections 3.2, 5, 4.2, and 4.2 respectively.

Four of the methods discussed in this section have achieved real-time performance. Two methods have incorporated cinematic idioms using constraints. One method is implemented in a static environment and the remaining methods are implemented in dynamic environments. The dynamic methods have incorporated constraint relaxation, frame coherence, and/or constraint weighting.

This section has dealt with camera control systems in 3D computer games. Constraints can be defined on scene objects and/or images. The underlying constraint solvers will find appropriate camera parameters for the shots which satisfy partially or fully the defined constraints. Comparison of the methods covered in this section is given in table 3.

Paper	Description	Comments
(Drucker and Zeltzer 1994)	Proposed a methodology to implement a constraint based framework which is a network of camera modules for intelligent camera control; Task analysis is to be done for a specific environment; Tasks are to be encapsulated into camera modules; Camera modules encapsulate user control, constraints, branching condition between the modules; The constraint solver finds camera parameters for the active camera module satisfying all the constraints of the module; Various interfaces are to be connected into the framework; Deals with static 3D environment	Achieve real-time performance with lots of preprocessing; Camera control system of a virtual museum is implemented; Constraints on camera parameters are identified based on the task analysis; Described four constraints; A generic camera module is consists of an initializer, controller, constraint list, and state components; The path planning camera control module for automatic navigation through the museum is described
(Bares and Lester 1999)	Developed visualization interface called CONSTRAINTCAM for dynamic 3D environment; The constraint solver will try to find a solution satisfying all the constraints; If needed, it will find solution in a single shot by relaxing weak constraints, in multiple shots, or in a composite shot with insets	A real-time application; Implemented a interactive testbed representing a virtual cityscape; The system supports four constraints; Constraint relaxation is supported; Constraint weighting is supported; Visibility is handled by the "occlusion avoidance" constraint;
(Halper, Helbing, and Strothotte 2001)	Used predictive camera planning; Each constraint has an optimal setting, a tolerance region, and relaxation parameter; The constraint solver find solution satisfying constraints in hierarchal order with subsequent constraint solutions has minimal effect on the previous ones; Implemented an exploration template	Claimed to be first frame coherent constraint-base camera engine; Tested the exploration template using three scenario; The system supports seven constraints; Constraint relaxation is supported; Occlusion avoidance is done using potential visibility regions techniques
(Lin, Shih, and Tsai 2004)	The camera control system has two components: description of shots and camera modules; Description of shots are finite state machines to capture scenes as per cinematic idioms; Camera modules represents a shot and encapsulate constraints	A real-time application; Incorporated cinematographic idioms using constraints; Frame coherence is supported; Constraint relaxation is supported; Supports occlusion detection and avoidance in some cases using predictive camera planning
(Bourne and Sattar 2005)	Each constraint has two parameters: a desired value and a weight; Cost of a constraint is difference of the desired and solution values; Total cost of solution is the sum of costs for all constraints; A stochastic local search based solver will find lowest cost solution for a frame	A real-time application; Implemented with changing or multiple targets; The system supports four constraints; Frame coherence is supported; Constraint weighting is supported; Visibility is maintained by using ray casting from the target

Table 3: Comparison of the methods

4. CAMERA CONTROL IN DEVOLEPMENT TOOLS

This section covers seven methods and provides a brief overview of the camera control systems used in development tools. Solutions of camera control problems for 3D computer games and animations are derived in the tools using various constraint sets and application of CSP solution techniques. The tools are used for generation of camera movement set, shots, and camera paths. In addition, application of cinematic idioms in 3D computer games using constraints and real-time performance of the methods are discussed.

In section 4.1, generation of the camera movement set is discussed. Camera shot planning is discussed in section 4.2. In section 4.3, camera path planning is discussed.

4.1. Camera Movement Set

The authors in (Jardillier and Languénou 1998) describe a virtual cameraman. The virtual cameraman generates a set of camera movements in a dynamic environment which satisfies the constraints. Then the user can select camera movements to create a desired animation. An alternative to the virtual cameraman is the approach which the authors call generation-and-test method. The alternative is a time consuming approach and unlikely to create desired animation at reasonable costs. The solution technique of the virtual cameraman is global instead of per frame basis. It does not use key framing. The solution set is for the complete animation.

The authors in (Jardillier and Languénou 1998) use a hard constraint solving strategy and pure interval solvers (Christie et al 2005).

The problem addressed in (Jardillier and Languénou 1998) is how to help graphic designers create camera movements at the same time avoiding all the technical details. They begin by referring to the ‘Declarative modelling’ approach of the geometric modeling community with three phases namely description, generation, and result exploration and state that the present work follows that approach. A user of the method can define constraints either on the image space or on the object of the scene or on both.

Constraints on the camera are:

1. “fixed location panoramic shot”.
2. “pure traveling”.
3. “high angle”.
4. “low angle” etc.

Constraints on the image space are:

1. “fully included in”.
2. “partially included in”.
3. “fully excluded of” etc.

Constraints on the object of the scene are:

1. “orientation of the object axis”.
2. “upside down”.
3. “stand up”.
4. “front side visible”.
5. “closer than” etc.

Time is also considered as a variable. Thus constraints can be defined for any arbitrary duration of animation. A global solving process computes a solution set of camera movements which satisfies multiple constraints for the complete animation. The constraint solver is based on interval arithmetic. It recursively evaluate the current interval function on hyper-rectangles using tri-valued logic. If evaluating response for a hyper-rectangle is true, it is a solution and will be added to the solution set. If evaluating response for a hyper-rectangle is false, sub hyper-rectangles of the hyper-rectangle will not be evaluated further. If evaluating response for a hyper-rectangle is unknown, the hyper-rectangle is subdivided into two sub hyper-rectangles which will be evaluated further similarly.

The authors describe mathematical models for the scene objects, camera, and their movements. Then constraints, constraint solver and three ways to include time variable namely ‘brute force’, ‘width first’, and ‘deep first’ methods are described. The authors state to have conducted an experiment. The test environment was a Dec Alpha 250 computer. The time required to obtain a solution set for a static and a dynamic camera problem was reported as 1 hour 30 minutes and 10 hours respectively. The authors claim that it is the first method that can compute a complete set of camera positioning solutions, the user given constraints are guaranteed to be satisfied because no key framing or interpolation is used, and need no external solver to implement the method.

The authors identify the following future work:

- To provide a way to present solution sets and navigate through it intelligently.
- To obtain solution set in near real time.
- To examine if visibility testing is required for the method instead of the Z-testing currently employed.

The results of the paper have been referred to by (Languénou et al 1998), (Christie, Languénou, and Granvilliers 2002), (Christie and Languénou 2003), and (Benhamou et al 2004).

The authors in (Languénou et al 1998) have dealt with the extension of the method of (Jardillier and Languénou 1998) which uses a basic interval solver based on recursive subdivision of the search space. The focus of the paper was on the use of more sophisticated constraint solving techniques. The method was implemented using high level programming in a CLP language. The user of the method can devise new image space constraints easily. Like the previous method of (Jardillier and Languénou 1998), the solution technique of the present method is global instead of per frame basis, does not

use key framing, the solution set is for the complete animation, and will satisfy multiple constraints.

The problem addressed in (Languénou et al 1998) is how to help graphic artists create camera movements easily and relieve them from the need of understanding mathematical concepts underling the camera movements. The authors begin by referring to their first implementation of the work in (Jardillier and Languénou 1998) which used an interval solver based on recursive subdivision of the search space and state that the first implementation of the method was not efficient enough to tackle complex camera movements. Like a user of the first implementation, a user of the present implementation of the method will be able to define constraints on the image space, on the objects of the scene, or on the both. The authors used Declarative Language with Interval Constraints (Declic) which utilized two solvers for enforcing hull-consistency over some primitive constraints and box-consistency over global complex constraints. The more sophisticated constraint solving techniques are employed to handle non-linear constraints aroused from modeling of camera movements and screen-space constraints. Interval computation is used in the solver to avoid key-framing and interpolation. An extension of the core local-consistency algorithm is used to process universally quantified time constraints. The solver will process global camera movements for the complete animation. It will generate a set of camera movement satisfying user defined constraints from which the users will be able to select camera movements to create desired animation.

The authors describe mathematical models of the scene objects, camera, and their movements. Constraints used in the paper are similar to the constraints used in (Jardillier and Languénou 1998). They also describe some of the constraints on the camera, the projected objects, and objects of the scene in Declic. The camera control method was under implementation. A prototype was implemented in Declic. The authors state that the users can devise new image space constraints easily using the approach and satisfaction of the specified constraints can be characterized precisely. They claim that the final implementation will minimize amount of repetitive tasks for the users and will require less image representation model knowledge of the users.

The authors identify the following future work:

- To improve the resolution process to increase number of degree of freedom of the camera.
- To use a high level constraint-oriented language for flexibility of development and easy maintenance.
- To devise a proper way to deal with constraints that must hold in a given interval as a built-in process in Declic.
- To demonstrate usability of the tool for the users having no knowledge of camera movement mathematics.

The authors in (Benhamou et al 2004) have discussed solutions of control and motion planning problems. The authors propose an alternative solution to a classical one based on sound numerical methods. These methods use interval arithmetic and enforce local consistency to prune search space. The presentation of the alternative solution technique

includes several relevant algorithms along with a few concepts and operators. Then application of the alternative solution technique to the camera control problems is discussed. Like (Jardillier and Languénou 1998) and (Languénou et al 1998), it generates a set of camera movement satisfying user defined constraints. The user can select camera movements from the set to create desired animation. The time required to find a solution set using the method is much less than the time required by the method described in (Jardillier and Languénou 1998).

The problem addressed in (Benhamou et al 2004) is how to find an efficient solution of control and motion planning problems and enable a graphic artist to specify the desired camera movements for a shot using cinematographic idioms. The authors observe that many control and motion planning problems can be reduced. The reduced problems become either to find a sound approximate solution space determined by a set of nonlinear equalities or the guaranteed tuning problem. In the guaranteed tuning problem reduction method, a value for some tuning parameter is found in order to verify a set of inequalities for all possible values of some perturbation vector. The classical approaches to solve these problems are based on quantifier elimination techniques. The constraint solver used for the present work utilized interval arithmetic and local consistency enforcement based sound numerical methods to prune the search space. Each constraint is considered one by one along with the sets of elements verifying all the constraints considered so far.

The authors present and prove soundness for the inner contracting operator based inner propagation algorithm (IPA) and two inner contracting operator algorithms (ICO1 and ICO2). The method IPABC is the algorithm IPA using ICO2 which in turn uses 'Outer-Box Contracting Operator'. They also present, analyze, and prove various concepts and operators relevant to the algorithms. They describe interval constraint solving using local consistencies. They investigate application of their method to camera control problems. The user defined descriptions for a shot are translated into constraints in terms of camera parameters. Constraints used in the paper are similar to the constraints used in (Jardillier and Languénou 1998) and (Languénou et al 1998). The authors state to have conducted experiments with the help of a prototype of a declarative modeler which used cinematic languages to specify camera motion. The prototype was implemented in C++ and Tcl/Tk. The test environment was a Pentium III, 800 MHz Linux PC. To generate all solutions for the nine benchmark problems described in the paper, IPABC method was 6.2 to 283 times faster than JLA method of (Jardillier and Languénou 1998). The authors observe that the efficiency of their method increased steadily with progressively higher precision. They claim that their method is much faster than both the soundly solving system of nonlinear inequalities and the guaranteed tuning problem with one dimensional perturbation vector.

The authors identify the following future work:

- To speed up interval constraint-based solvers.
- To allow camera movement along all the degrees of freedom with reduced number of variables.

- To compare inner approximation of relations computation using “Kaucher arithmetic” and using outer contracting operators.

4.2. Camera Shot Planning

The authors of (Bares et al 2000) observe that a constraint based virtual camera system should have the following elements/capabilities:

- A virtual constraint editor.
- Capable of environmental analysis.
- Able to evaluate shot quality.
- Have real time interactive performance.

The storyboard frames are used by a director of a film to define a desired shot. The Frame Editor described in (Bares et al 2000) can be used to draw storyboard frames. The user like a director of a film can use the editor to specify how objects of interest should be displayed. The constraints solver like the cinematographer of a film will find camera parameters satisfying all the constraints derived from the storyboard frames to display the shot. The constraints used in the paper are based on established practices in photography and cinematography to get proper photographic composition. Photographic composition is defined as an artful arrangement of visual elements like subject size and location, view angle, occlusion, exclusion, and depth.

The environment in (Bares et al 2000) is a dynamic one. The solution technique is an in-complete search method and can not handle constraint failures. The authors have used heuristic search based solvers (Christie et al 2005).

The problem addressed in (Bares et al 2000) is how to define camera compositions efficiently avoiding both pre-defined camera placements and script generated constraints in 3D environments. The authors propose a system consisting of a graphical interface called the Storyboard Frame Editor and a recursive heuristic constraint solver. The frames drawn on the editor are automatically encoded into a set of constraints. The editor export files in two forms of shot definition namely the relative displacement definition and the constraint definition. The relative displacement definition is used by the solver for an instantaneous solution if configuration of scene objects match with those of the editor’s 3D scene model. Otherwise, the solver use exhaustive generation and test method. At first it finds a valid region for each constraint. Then it examines candidate camera shots by discretely incrementing camera parameters inside the respective valid regions. Each camera placement is evaluated with the help of individual constraint evaluation function to find individual constraint satisfaction rating. The cumulative constraint satisfaction rating is the weighted sum of the individual constraint satisfaction ratings. The search begins at relatively coarse resolution candidate shots to find pre-defined numbers of best candidate shots based on cumulative satisfaction ratings. The search space around the best candidate shots are recursively searched with increasingly refined resolution. If no camera position is found which satisfies all the constraints, the solver reports failure.

Fifteen constraints are supported by the method including the following constraints:

1. “CAM_POS_IN_REGION”.
2. “OBJ_EXCLUDE_OR_HIDE”.
3. “OBJ_IN_FIELD_OF_VIEW”.
4. “OBJ_OCCLUSION_MINIMIZE”.
5. “OBJ_OCCLUSION_PARTIAL”.
6. “OBJ_PROJECTION_ABSOLUTE”.
7. “OBJ_PROJECTION_SIZE”.
8. “OBJ_VIEW_ANGLE” etc.

The authors describe functionality of the Storyboard Frame Editor, important constraints listed above, and the heuristic constraint solver algorithm. The constraints used in the paper are similar to the constraints used in (Bares, Thainimit, and McDermott 2000). The test environment was an Intergraph GL2 computer with Pentium II 400MHz processor, 256 MB RAM, and VX113 AGP OpenGL accelerator. Four camera shot problems are explained and their solution shots are shown in the figures 10-14 of the paper. The cumulative constraint satisfaction rating of the four shots are shown in the table 4 based on two implementations of the constraint solver. The authors claim that the Storyboard Frame Editor and the constraint solver together is a tool to define how the virtual camera should film objects of interest without worrying about the placement of the camera.

Figure	*Heuristic Search*	*Relative Displacement*
10	0.877109	0.0
11	0.850244	0.338701
12	0.774565	0.338701
14	0.810209	0.473219

Table 4: Cumulative Constraint Satisfaction Rating for Four Solution Shots given in (Bares et al 2000; page 186; table 3)

The authors identify the following future work:

- To optimize the search heuristic and constraint evaluators to obtained real-time performance.
- The constraint solver module is to be added with several prototypes of interactive 3D virtual environments.
- To add additional constraint types by coding new evaluators and script parsing functions.
- To add capability to handle failure to find a constraint solution, either less important constraint will be relaxed or multiple shots visualization will be used.
- To analyze effectiveness of the user interface and aesthetics of the computed shots.

As described in (Bares, Thainimit, and McDermott 2000), a director specifies a shot using storyboard frames or notes in the film industry. Placement of the camera is handled by a cinematographer based on the storyboard frames or the director’s notes. Camera planning problems cast as constraint satisfaction problem has natural correspondence to the actual practices in cinema creation. A set of constraints on camera or objects in the scene, or on the both can define a camera shot. Then a constraint solver finds a solution

shot. A constraint solution is an assignment of values to the camera parameters. The camera shot is composed by satisfying constraints on how each subject appears in the frame. The authors state that camera placement in virtual environment can be defined by the following vectors:

- “Camera position vector”.
- “Aim direction vector”.
- “Field of view angles”.
- “Up orientation vector parameters”.

The following design guidelines for general purpose camera constraint system are described in (Bares, Thainimit, and McDermott 2000):

- “Arbitrary viewing goals”.
- “Environmental complexity”.
- “World non-interference”.
- “Failure handling”.

The problem addressed in (Bares, Thainimit, and McDermott 2000) is how to plan camera shots automatically in virtual 3D environments such that each shot will communicate some specific visual message or goal. The authors begin by referring to their previous work in (Bares, Grégoire, and Lester 1998) and (Bares and Lester 1999) and state that those work served as a starting point for this paper. The authors propose a constraint based camera planning method for 3D virtual environments. A user or a software system requests to visualize subjects of interest and how they should be viewed. The view request will be translated by constraint script into a set of constraints. The constraint solver tries to find a solution camera shot based on given constraints. It uses an exhaustive generate-and-test method to determine the camera placement having the highest cumulative constraint satisfaction rating. The cumulative constraint satisfaction rating is weighted sum of all individual constraint satisfaction measured as product of relative priority and satisfaction rating of the constraint.

Fifteen constraints are supported by the method including the following constraints:

1. “CAM_FIELD_OF_VIEW”.
2. “CAM_POS_IN_REGION”.
3. “LOOK_AT_POINT”.
4. “OBJ_DEPTH_ORDER”.
5. “OBJ_DISTANCE”.
6. “OBJ_IN_FIELD_OF_VIEW”.
7. “OBJ_OCCLUSION_MINIMIZE”.
8. “OBJ_PROJECTION_ABSOLUTE”.
9. “OBJ_PROJECTION_RELATIVE”.
10. “OBJ_PROJECTION_SIZE”.
11. “OBJ_VIEW_ANGLE” etc.

The authors describe the method and define important constraints listed above. The constraints used in the paper are similar to the constraints used in (Bares et al 2000). An example application of the method to find a camera position for an over the shoulder shot

during a conversation between two players is explained. The test environment was a Pentium II 400MHz processor. The constraint solver searched camera position over a 20x20x20 grid. The aim direction was at 15° interval and field of view angle was over a range of 10 values. The authors report that a total of 13,891,500 shots were evaluated in approximately 30 minutes to determine the solution shot with 89% satisfaction rating. The virtual camera constraint model described in the paper was under development. The authors claim that the constraints can be used to create cinematic camera shots and to communicate specific visual message.

The authors identify the following future work:

- To develop a more efficient constraint solution search algorithm.
- To develop a WYSIWYG graphic interface to generate automatically constraint script for a desired camera shot.

The results of the paper have been referred to by (Lin, Shih, and Tsai 2004).

4.3. Camera Path Planning

A high level modeling approach to camera control is introduced in (Christie, Languénou, and Granvilliers 2002). Two aims of the camera control method are described in the paper as to find camera parameters in order to ease the camera path creation and to give the user a declarative tool for animation modeling.

As stated in (Christie, Languénou, and Granvilliers 2002), seven degrees of freedom of camera parameters are based on translation, rotation, and focus. Each property of the desired image is defined by a set of constraints over all the camera parameters based on established cinematographic techniques. Solution of the constraint problem should be done per frame basis for animation. Hence classical constraint solution techniques such as propagation can not be applied directly with reasonable cost. To find a camera path to visualize a shot, a dynamic system using a complete search method based on interval filtering is used. A solution satisfies all the constraints.

The problem addressed in (Christie, Languénou, and Granvilliers 2002) is how to determine a camera path according to user given declarative properties such as locations or orientations of objects on the screen at a given time. The authors begin by referring to the approach of (Jardillier and Languénou 1998) and state that they have followed that approach. The authors propose a constraint based camera control method to find camera paths based on cinematographic techniques. The paths are based on user defined sequences of parameterized elementary camera movements. These cinematic primitives are called hypertubes. The hypertubes are defined for linear movement (traveling), rotation around the camera's horizontal or vertical axes (panoramic), zoom-in, zoom-out, and arcing to turn around objects. The hypertubes are connected by intertubes which are relations between two successive hypertubes that guarantee continuity of camera movement. A hypertube with a set of properties and duration is a Numeric CSP (NCSP). Each cinematographic property is redefined as constraints on the hypertube variables. To model a shot, a user like a director of a film provides properties like orientations or

locations of the objects on the screen and a sequence of elementary camera movements. Then interval consistency and quantified constraint based solving algorithms will compute parameters of the hypertubes. The search algorithm combines constraint propagation and enumeration to compute a solution of the global NCSP consists of individual hypertube NCSPs. The solver uses interval-based filtering to remove inconsistencies and isolate inner boxes for each hypertubes. It finds a solution based on optimization criterion from the set of the isolated inner boxes of the given hypertube in order to determine starting conditions for the next hypertube. It uses Tabu strategy based backtracking to remove inconsistent hypertubes.

The authors describe the hypertube model and the search algorithm. The OpAC C++ library with an extension to manage universally quantified constraints is used to implement the constraint solving algorithm. The test environment was a Pentium II, 350MHz Linux PC. The authors state to have conducted experiments. The numerical results obtained are shown in the table 5 for the three examples. The calculated camera path for Example #2 of the table 5 is shown in the figure 2. The 3D scene for Example #2 has a static object A and a moving object B. The shot consists of three basic movements: a horizontal panoramic from A to B, an arching around B, and another horizontal panoramic to follow B. The execution time reported for Example #2 is around 5 seconds for a 6 seconds animation. The authors observe that time required by the approach of the paper to generate camera paths is around 4 to 10 seconds for the problems defined with 18 to 48 variables in comparison to the time needed by the earlier approach of (Jardillier and Languénou 1998) is more than 10 minutes for the problem defined with more than 7 variables only (both for the first solution). The authors claim that their approach shows a great improvement in time and animation quality over the former approaches and the encouraging experimental test results open exciting perspectives to apply it for visual occlusion handling etc.

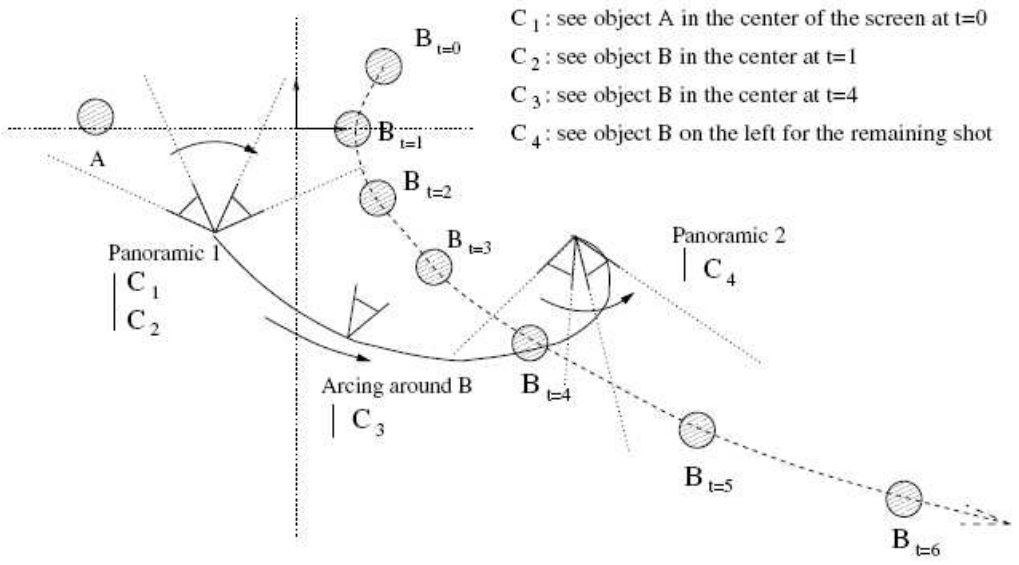

Figure 2: Calculated Camera Path for Example #2 (Christie, Languénou, and Granvilliers, 2002, page 629, Fig. 11)

Problem	Hypertubes	Variables	Constraints	First solution (ms)
Example #1	2	18	9	3940
Example #2	3	22	12	5290
Example #3	6	48	24	9880

Table 5: Computation Time for Three Problems (Christie, Languénou, and Granvilliers, 2002, page 629, table 2)

The authors identify the following future work:

- To find the closest solution to the user modification of the location or orientation of objects in the 3D space.
- To generate the hypertubes on the fly considering events in the 3D scene.
- To search and implement other constraint satisfaction problems expressed by means of CSP decomposition and variable linking.

The hypertube model to compute camera path for animation is also used in (Christie and Languénou 2003) to avoid expensive computation, camera movement restrictions, and poor interactivity. It is a CSP approach for camera control based on established cinematographic techniques. It provides a global control over the camera path. It has fine control over the image using cinematographic properties.

It seems that the paper (Christie and Languénou 2003) is another version of the paper by (Christie, Languénou, and Granvilliers 2002) using similar techniques.

The problem addressed in (Christie and Languénou 2003) is how to determine camera path according to user given declarative properties such as locations or orientations of objects on the screen at a given time. The authors begin by referring to the approach of (Jardillier and Languénou 1998) and state that they have followed that approach. The authors propose a constraint based camera control method to find camera paths based on cinematographic techniques. The paths are based on user defined sequences of parameterized elementary camera movements called hypertubes. The search algorithm combines constraint propagation and enumeration to compute a solution of the global NCSP consists of individual hypertube NCSPs. The solver uses interval-based filtering to remove areas having no solutions for a hypertube and backtracking to remove inconsistent hypertubes.

The authors describe the hypertube model and the search algorithm. A C++ based prototype was used for implementation of the model. The test environment was a Pentium III, 800MHz Linux PC. The authors state to have conducted experiments. The results are shown in the table 6 for two examples. The scenes for the test consist of mobile objects A and B, and a static object C. In addition, the scene of example #2 has obstacles shown as red circles. Camera paths are calculated for a traveling, a local arching, and panoramic camera movements. The paths are shown in top and bottom part of figure 3 for Example #1 and Example #2 respectively. The authors observe that time required by the approach of the paper to generate camera paths is around 3 to 6 seconds for problems defined with 26 variables in comparison to the time needed by the earlier approach of (Jardillier and Languénou 1998) is more than 10 minutes for the problem

defined with more than 7 variables only (both for the first solution). The test results are reported as very encouraging and as a consequence open up the way for real-time application of the approach. The authors claim that the approach fine tunes the images in the shots following cinematographic idioms and also overcomes expensive computations, restricted movements, and poor interactivity problems of the approach of (Jardillier and Languénou 1998).

Problem	Hypertubes	Variables	Constraints	First solution (ms)
Example #1	3	26	17	3170
Example #2	3	26	25	6090

Table 6: Computation Time for Two Problems (Christie and Languénou 2003; page 9; table 4)

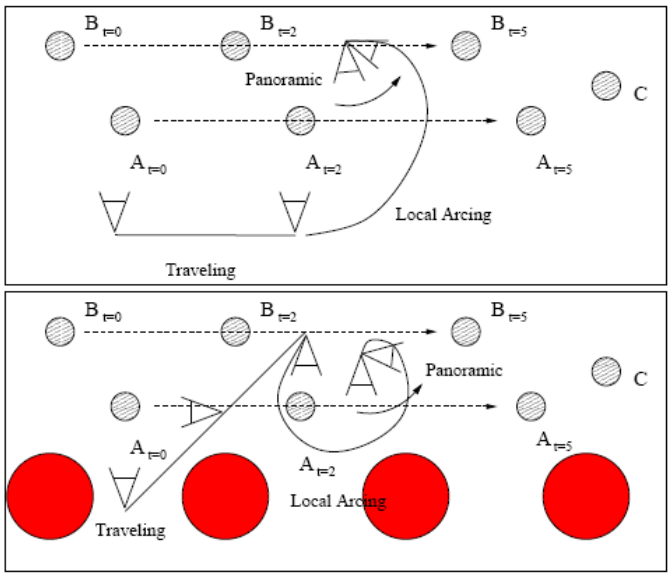

Figure 3: Calculated Camera Path (Christie and Languénou 2003; page 9; Fig. 4)

The authors identify future work as to generation of the hypertubes on the fly based on the events generated in the 3D scene.

This section has covered seven methods of camera control used in development tools for 3D games. Three of the methods discussed in the section deal with generation of the camera movement set which can be used to create animation. Two methods deal with individual camera shot planning. Two methods deal with camera path planning for animation.

This section has dealt with camera planning in development tools of 3D computer games. With the help of the development tools, the users can develop their desired shot and/or complete animation by specifying constraints. Underlying constraint solvers find appropriate camera parameters for the shot and/or the complete animation, which satisfy user specifications. Comparison of the methods covered in the section is given in table 7.

Paper	Description	Comments
(Jardillier and Languénou 1998)	Supports constraints on the camera, on the image space, and on the object of the space; The constraint solver is based on interval arithmetic which recursively evaluate the current interval function on hyper-rectangles using tri-valued logic; The method computes a solution set of camera movements which satisfies multiple constraints for the complete animation	The authors believe that this is the first method that computes a solution set for camera positioning; Users can select camera movements from the set to create a desired animation; Z-testing is used for occlusion avoidance
(Languénou et al 1998)	Supports constraints on the image space, on the objects of the scene, or on the both; Users can devise new image space constraints easily; An extension of the core local-consistency algorithm is used to process universally quantified time constraint; Like (Jardillier and Languénou 1998) the method also computes a solution set of camera movements which satisfies multiple constraints for the complete animation	Users can select camera movements from the set to create a desired animation; Provides more sophisticated solving techniques which can handle non-linear constraints aroused from modeling of camera movements and screen space constraints; Satisfaction of the specific constraints can be characterized precisely; Cinematographic techniques are supported
(Bares et al 2000)	The Frame Editor can be used to draw storyboard frames; The user like a director of a film can use the editor to specify how objects of interest should be displayed; Constraints are derived from the storyboard frames automatically; Constraints supported in the paper are similar to the constraints used in (Bares, Thainimit, and McDermott 2000); The constraints solver like cinematographer of a film will find camera parameter satisfying all the constraints to display the shot	Deals with camera shot planning; Supports fifteen constraints; Cinematographic techniques are supported; Constraint weighting for relative priority is supported; Visibility is deals with occlusion constraint which is evaluated by either ray casting from the camera or frame rendering of both the object of interest and potential occlusion
(Bares, Thainimit, and McDermott 2000)	Constraints supported in the paper are similar to the constraints used in (Bares et al 2000); The solver finds a solution camera shot with highest cumulative constraint satisfaction rating using generation and test method	Deals with camera shot planning; Supports fifteen constraints; Cinematographic techniques are supported; Constraint weighting for relative priority are supported
(Christie, Languénou, and Granvilliers 2002)	The camera path is defined as sequence of parameterized elementary movements called hypertubes; Cinematographic properties are defined as constraints on hypertube variables; The search algorithm combines constraint propagation and enumeration to compute a solution of the global NCSP consists of individual hypertube NCSP	A declarative tool for animation modeling; A hypertube with a set of properties and duration is a NCSP; Cinematographic techniques are supported
(Christie and Languénou 2003)	Same as (Christie, Languénou, and Granvilliers 2002)	Same as (Christie, Languénou, and Granvilliers 2002)

(Benhamou et al 2004)	Constraints supported are similar to the constraints used in (Jardillier and Languénou 1998) and (Languénou et al 1998); The constraint solver utilized interval arithmetic and local consistency enforcement based sound numerical methods to prune the search space; Like (Jardillier and Languénou 1998) and (Languénou et al 1998) the method also computes a solution set of camera movements which satisfies multiple constraints for the complete animation	User can select camera movements from the set to create a desired animation; Average time required to generate a camera movement set is less than (Jardillier and Languénou 1998); Cinematographic techniques are supported

Table 7: Comparison of the methods

5. CAMERA CONTROL IN INTERACTIVE NARRATIVES

This section covers two methods and provides a brief overview of the camera control systems used in 3D interactive narrative and storytelling. The two methods considered in this section also implement cinematographic rules found in the film making to the 3D virtual environment. One method constructs a scene tree and desired camera location is searched using keywords. The other method uses film idioms as a plan operator.

The authors in (Amerson and Kime 2000) have dealt with automated cinematography in the context of interactive narratives in a 3D virtual world. Cinematographic idioms are defined as constraints to find camera placement for any shot. Due to utilization of film idioms, the camera position and movement convey narrative information along with the events of the narratives.

The exact flow of action is not predefined in interactive narratives. The user can substantially change flow of action at any time. The flow of action in the interactive narratives can only be predicted with high degree of uncertainty. The authors observe in (Amerson and Kime 2000) that due to the uncertainty an automated camera control system in interactive environment can not utilize all the idioms of cinematography.

The problem addressed in (Amerson and Kime 2000) is how to apply the cinematographic techniques in the camera control system of the 3D virtual world interactive narratives. The authors begin by referring to the works of (Christianson et al 1996) and state that the camera control concept based on cinematographic constraints of that paper is the foundation of the present work. Then they refer to the works of (He, Cohen, and Salesin 1996) and state that a scene hierarchy to represent cinematography conventions used in that paper is the basis to encode common film idioms into a scene tree. At last they refer to a constraint-based approach called UCAM described in (Bares and Lester 1997) and state that the method of the present paper is a hybrid system based on the three above mentioned methods. The authors propose a real-time camera control system called Film Idiom Language and Model (FILM) that uses cinematographic techniques and inputs from a Narrative Planner to constraint the location and orientation of the camera in a virtual world of the interactive narratives. The film idioms are encoded in scene level. A scene tree is constructed using FILM language. A keyword list is used to search the tree for shot selection. The Narrative Planner generates information based on the actions carried out by the characters and outcome of these actions. This information is translated by a domain specific Translator. A domain independent Director uses the translated scene requirements to select a film idiom from the scene tree. It bounds any unbounded variables in the scene to the virtual world objects. These scene specifications along with the constraints are used by a graphics engine specific Cinematographer to choose the best camera position to satisfy these constraints. If the selected optimal position is not a valid position, constraints are relaxed based on the weights defined in the scene specification using an ad hoc procedural model to find a closest point to the optimal position which satisfy the remaining constraints. The following constraints are used by the method:

1. “lensType”.

2. “lookAt”.
3. “relativeLocation”.
4. “maintainLensType”.
5. “maintainRotation”.
6. “maintainAbsoluteLocation” etc.

The authors explain the camera control architecture in terms of its components: the scene tree, the FILM language, the translator, the director, and the cinematographer. The camera control system was implemented as a part of the Mimesis project. A modified Unreal Tournament (UT) game engine was used as the first development platform. The AI server of the Mimesis system communicates with the modified UT game server. The AI server consists of a HTN-style decomposition planner called Longbow and other supporting subsystems. Longbow formulates plans. The UT server passes user actions to AI server as well as executes the plans formulated by the AI server. The camera control approach was under implementation. The authors claim that after implementation of their method, it will be able to select abstract film idiom knowledge and convert that knowledge into virtual camera placements.

The authors identify the future work as to use a more general constraint solver.

The authors observe in (Jhala and Young 2005) that existing camera systems in virtual 3D environments do not “exploit narrative structure and causal relationship between shot or scene segments” to select camera positions. The existing systems only try to find best framing and smooth transition of shots. As narrative based virtual environment become more complex the need for effective communication of information increases. A user perceives the virtual environment through a virtual camera. The camera can be considered a powerful tool of communication for conveying events, mood of the scene, pace or tempo of the underlying narrative, and relationship between objects within virtual environment.

The problem addressed in (Jhala and Young 2005) is how to communicate effective information to the users of complex narrative-based virtual environments using camera control. The authors propose a camera planning system for narratives in virtual environments where decisions are made in three levels: selection of cinematic geometric composition, camera parameter for information communication, and camera shots and transition to maintain rhetoric coherence. The camera planning system is similar to the film production pipeline. The film idioms are formalized to communicate effective information and represented as plan operators. The representation of idioms has casual motivation and hierarchy. It allows the system designer to rank candidate shot sequences. The story is represented as a sequence of actions executed in the 3D world. The knowledge base of the discourse camera planner consists of the sequence of action executed in the 3D world, and the annotations indicating properties and characters of the story. The planning system incrementally formulates plans by adding steps to the existing plan to satisfy a goal state or a pre-condition of an existing action by the effect of the action, or to refine an abstract action by expansion. It uses causal reasoning, temporal reasoning, and decomposition during addition of steps. Then a ranking function with user

preference chooses the best plan. The discourse camera planner utilizes a library of hierarchical plan operators to generate camera directive sequence for specific type of action sequences. The camera directives are translated into constraints. Then a geometric constraint solver implemented in the underlying game engine will find a camera placement solution which satisfies the constraints.

The authors describe the formalization of film idioms. The approach was implemented using a service-oriented architecture called Mimesis for the control of narratives in a virtual world. The proposed camera planning algorithm uses a de-compositional partial-order casual link planning system called Longbow which is implemented in LISP. The camera control was tested within the Unreal Tournament 2003 game engine. The authors observe that the approach may not work at real time for large stories because it needs the story details for pre-planning camera movement. They claim that they have demonstrated how the effective elements of a story are communicated by generation of communicative plans by a discourse planning algorithm through a plan space search.

The authors identify the following future work:

- To generate automatically the annotations of the story plan with additional information about the action and characters from knowledge of plan structure.
- To optimize geometric constraints solver by utilizing high level context aware directives to reduce search space.
- To develop strategies for comparative empirical evaluation of cinematic camera planning systems.

Paper	Description	Comments
(Amerson and Kime 2000)	A scene tree is constructed using FILM and keyword list is used to search the tree for shot selection; A Narrative Planner generates information based on the actions carried out by the characters; A Director uses translated scene requirements to select a film idiom from the scene tree; The scene specifications along with the constraints are used by a Cinematographer to choose the best camera position to satisfy these constraints	Proposed a real-time camera control system; Film Idiom Language and Model (FILM) developed; Cinematic idioms are incorporated; Constraint relaxation is supported; Constraint weighting is supported
(Jhala and Young 2005)	The film idioms are formalized to communicate effective information and represented as plan operators; The planning system incrementally formulates plans; Then a ranking function with user preference is used to choose the best plan; The discourse camera planner utilizes the plan operators to generate camera directives; The camera directives are translated into constraints; Then a geometric constraint solver will find a camera placement solution which satisfies the constraints	Proposed a real-time camera planning system for narratives in virtual environments; Cinematic idioms are incorporated

Table 8: Comparison of the methods

This section has covered two methods of camera control in interactive narratives. Both of the methods are real-time applications and included cinematic techniques. In addition, one method supports constraint relaxation and constraint weighting. As in camera control in 3D computer games, defining constraints and CSP solution techniques can also be applied in interactive narratives. Comparison of the methods covered in the section is given in table 8.

6. CONCLUSION

We have considered twenty papers in this review. Six background papers are concerned with constraints and their solving techniques (Nadel 1990; Kumar 1992; Badros 1998), and camera control systems (Hornung, Lakemeyer, and Trogemann 2003; Giors 2004; Christie et al 2005).

We have considered fourteen papers in the main body of the survey. The camera control methods of the papers handle the camera system in 3D computer games and narratives. One paper (Drucker and Zeltzer 1994) is concerned with visiting a virtual museum. Four papers (Bares and Lester 1999; Halper, Helbing, and Strothotte 2001; Lin, Shih, and Tsai 2004; Bourne and Sattar 2005) are concerned with other computer games. Three papers (Jardillier and Languénou 1998; Languénou et al 1998; Benhamou et al 2004) are concerned with providing set of camera movements to assist the user to create animation. Two papers (Bares et al 2000; Bares, Thainimit, and McDermott 2000) are concerned with individual shot planning. One of them (Bares et al 2000) is concerned with specifying constraints from storyboard frame. Two papers (Christie, Languénou, and Granvilliers 2002; Christie and Languénou 2003) are concerned with finding camera paths for animation. Two papers (Amerson and Kime 2000; Jhala and Young 2005) are concerned with interactive narratives.

Section	Paper	Constraints supported	Solvers
Camera control in Computer Games	(Drucker and Zeltzer 1994)	"maintain the camera's up vector to align with world up"; "maintain height relative to the floor"; "maintain the camera's gaze (i.e., view normal) toward a specified object"; "maintain the camera's position on a collision free path through world" etc.	Based on sequential quadratic programming technique
	(Bares and Lester 1999)	"Vantage Angle"; "Viewing Distance"; "Occlusion Avoidance"; "Room Enclosure"	Hierarchical constraint solver
	(Halper, Helbing, and Strothotte 2001)	"Height angle"; "Level at"; "Facing"; "Angle to line of interest"; "Size"; "Visibility"; "Look at"	Ad-hoc incremental solver
	(Lin, Shih, and Tsai 2004)	Same as in (Halper, Helbing, and Strothotte 2001)	Same as in (Halper, Helbing, and Strothotte 2001)
	(Bourne and Sattar 2005)	"Height"; "Distance"; "Orientation"; "Frame Coherence"	Stochastic local search

Camera control in Development Tools	(Jardillier and Languénou 1998)	"fixed location panoramic shot"; "pure traveling"; "high angle"; "low angle"; "fully included in"; "partially included in"; "fully excluded of"; "orientation of the object axis"; "upside down"; "stand up"; "front side visible"; "closer than" etc.	Pure interval arithmetic
	(Languénou et al 1998)	Same as in (Jardillier and Languénou, 1998)	Interval based solver capable to handle non-linear constraints
	(Bares et al 2000)	"CAM_POS_IN_REGION"; "OBJ_EXCLUDE_OR_HIDE"; "OBJ_IN_FIELD_OF_VIEW"; "OBJ_OCCLUSION_MINIMIZE"; "OBJ_OCCLUSION_PARTIAL"; "OBJ_PROJECTION_ABSOLUTE"; "OBJ_PROJECTION_SIZE"; "OBJ_VIEW_ANGLE" etc.	Recursive heuristic search solver
	(Bares, Thainimit, and McDermott 2000)	"CAM_FIELD_OF_VIEW"; "CAM_POS_IN_REGION"; "LOOK_AT_POINT"; "OBJ_IN_FIELD_OF_VIEW"; "OBJ_DEPTH_ORDER"; "OBJ_DISTANCE"; "OBJ_OCCLUSION_MINIMIZE"; "OBJ_PROJECTION_ABSOLUTE"; "OBJ_PROJECTION_RELATIVE"; "OBJ_PROJECTION_SIZE"; "OBJ_VIEW_ANGLE" etc.	Generate-and-test
	(Christie, Languénou, and Granvilliers 2002)	Constraints based on cinematographic properties are on screen space such as location, orientation etc. of the objects	Interval consistency and quantified constraint solver; Used Tabu strategy based backtracking
	(Christie and Languénou 2003)	Similar to (Christie, Languénou, and Granvilliers 2002)	Similar to (Christie, Languénou, and Granvilliers 2002)
	(Benhamou et al 2004)	Same as in (Jardillier and Languénou 1998) and (Languénou et al 1998)	Interval arithmetic based solver with enforced local consistency to prune search space

Camera control in Interactive Narratives	(Amerson and Kime 2000)	"lensType"; "lookAt"; "relativeLocation"; "maintainLensType"; "maintainRotation"; "maintainAbsoluteLocation" etc.	Procedural model for relaxation of constraints
	(Jhala and Young 2005)	Film idioms are represented with constraints at scene and shot levels	Geometric constraint solver

Table 9: List of constraints and solvers used for camera control systems of fourteen papers included in the main body of the survey

Constraints are defined on-camera, on image space, on objects of the scene. Constraints related with camera path planning are defined too. All the constraints described in the papers are listed in the table 9.

Various CSP solution techniques are used by the camera control methods. Such as recursive heuristic search, generate-and-test, interval arithmetic based solvers, improved interval constraint solver, stochastic local search, geometric constraint solver etc. All the solution techniques described in the papers are also listed in the table 9.

The camera control method of (Drucker and Zeltzer 1994) has dealt with a static 3D environment. All other methods have dealt with dynamic 3D environments.

Six methods are real-time applications. Ten methods defined constraints to incorporate cinematic idioms into the 3D virtual environment. Three methods dealt with frame coherence. Four methods employed constraint relaxation method to find less than satisfactory solution in case of constraint failure. Five methods used weighting scheme to prioritize the constraints. Real-time performance and various techniques used in the methods of the papers in the main body of the review are listed in table 10.

All of the methods implemented third person perspective camera control system. It is observed that third person perspective camera system will enhance viewing experience of the user significantly (Lin, Shih, and Tsai 2004; Christie et al 2005). The camera setting of the camera system will communicate much more information than first person perspective camera system.

Application of CSP solution techniques to the camera control problem in computer game is relatively new phenomenon. Performance of the approaches does not justify deployment of the systems to the commercial computer games yet. On the other hand, the future directions of the research identified by the authors in the reviewed papers are many and diverse which indicate that it is a vibrant and active research field.

During the survey, we have come across works of many groups of active researchers in the field. There is high expectation among researchers in the field that successful

application of CSP solving technique to the camera control problem in the 3D commercial computer games will become feasible in the near future.

Section	Paper	Real-time performance	Cinematic supported	Frame coherence supported	Constraint relaxation supported	Constraint weighting supported
Camera control in Computer Games	(Drucker and Zeltzer 1994)	√				
	(Bares and Lester 1999)	√	√		√	√
	(Halper, Helbing, and Strothotte 2001)			√	√	
	(Lin, Shih, and Tsai 2004)	√	√	√	√	
	(Bourne and Sattar 2005)	√		√		√
Camera control in Development Tools	(Jardillier and Languénou 1998)					
	(Languénou et al 1998)		√			
	(Bares et al 2000)		√			√
	(Bares, Thainimit, and McDermott 2000)		√			√
	(Christie, Languénou, and Granvilliers 2002)		√			
	(Christie and Languénou 2003)		√			
	(Benhamou et al 2004)		√			
Camera control in Interactive Narratives	(Amerson and Kime 2000)	√	√		√	√
	(Jhala and Young 2005)	√	√			

Table 10: Real-time performance and various techniques of the camera control methods of fourteen papers included in the main body of the survey

ACKNOWLEDGMENT

I would like to thank my supervisor Dr. Scott Goodwin for his help, encouragement, discussion, and remarks. I would also like to thank Dr. Richard Frost for his valuable comments, suggestion, discussion, and guidance to complete the survey.

REFERENCES

1. Amerson, D. and Kime, S. (2000) Real Time Cinematic Camera Control for Interactive Narratives. *American Association for Artificial Intelligence, 2000*, 1-4.

2. Badler, N., Manoocherhri, K., and Baraff, D. (1986) Multi-dimentional Input Techniques and Articulated Figure Positioning by Multiple Constraints. In Proceedings of *the 1986 Workshop on Interactive 3D Graphic*s, 151-170.

3. Badros, G. J. (1998) *Constraints in Interactive Graphical Applications.* Ph.D. General Examination, Department of Computer Science and Engineering, University of Washington, Box 352350, Seattle, WA 98195-2350, USA.

4. Bares, W., Grégoire, J. P., and Lester, J. C. (1998) Realtime Constraint-based Cinematography for Complex Interactive 3D Worlds. In *IAAI-98: Proceedings of the Tenth National Conference on Innovative Applications of Artificial Intelligence,* Madison, Wisconsin, USA, 1101-1106.

5. Bares, W. H. and Kim, B. (2001) Generating Virtual Camera Compositions. In Proceedings of *the Sixth International Conference on Intelligent User Interfaces,* Santa Fe, New Mexico, USA, 9-12.

6. Bares, W. H. and Lester, J. C. (1997) Cinematographic User Models for Automated Real-time Camera Control in Dynamic 3D Environments. In Proceedings of *the Sixth International Conference on User Modeling,* 215-226.

7. Bares, W. H. and Lester, J. C. (1999) Intelligent Multi-shot Visualization Interfaces for Dynamic 3D Worlds. In *IUI'99: Proceedings of 1999 International Conference on Intelligent User Interfaces,* Los Angeles, California, USA, 119-126.

8. Bares, W., McDermott, S., Boudreaux, C., and Thainimit, S. (2000) Virtual 3D Camera Composition from Frame Constraints. In *Eight ACM International Conference on Multimedia,* Marina Del Ray, California, USA, 177-186.

9. Bares, W. H., Thainimit, S., and McDermott, S. (2000) A Model for Constraint-based Camera Planning. In *AAAI 2000 Spring Symposium Series on Smart Graphics,* Stanford, California, USA, 84-91.

10. Barr, A. H., Currin, B., Gabriel, S., and Hughes, J. F. (1992) Smooth Interpolation of Orientations with Angular Velocity Constraints using Quaternions. In Proceedings of *SIGGRAPH '92, Computer Graphics,* **26(2)** 313-320.

11. Benhamou, F., and Goualard, F. (2000) Universally Quantified Interval Constraints. In Dechter, R., editor, *Principles and Practice of Constraint Programming - CP 2000, Sixth International Conference Proceedings,* Singapore, LNCS, Springer, ISBN 3-540-41053-8, **1894** 67-82.

12. Benhamou, F., Goualard, F., Languénou, E., and Christie, M. (1999) An Algorithm to Compute Inner Approximations of Relations for Interval Constraints. *Ershov Memorial Conference 1999,* 416-423.

13. Benhamou, F., Goualard, F., Languénou, E., and Christie, M. (2004) Interval Constraint Solving for Camera Control and Motion Planning. *ACM Transactions on Computational Logic,* **5(4)** 732-767.

14. Blinn, J. (1998) Where am I? What am I Looking at? In *IEEE Computer Graphics and Applications, July1998,* 76-81.

15. Bourne, O. (2004) *Constraint-Based Intelligent Camera Control for Interactive Digital Entertainment,* Ph.D. thesis, Griffith University, Australia.

16. Bourne, O. and Sattar, A. (2004) Applying Constraint Satisfaction Techniques to 3D Camera Control. *Seventeenth Australian Joint Conference on Artificial Intelligence*, Cairns, Australia, 658-669.

17. Bourne, O. and Sattar, A. (2005) Applying Constraint Weighting to Autonomous Camera Control. In *Artificial Intelligence and Interactive Digital Entertainment*, Marina Del Ray, CA, USA, AAAI Press, 3-8.

18. Christianson, D. B., Anderson, S. E., He, L., Salesin, D., Weld, D. S., and Cohen, M. F. (1996) Declarative Camera Control for Automatic Cinematography. In Proceedings of the *AAAI-96: Proceedings of the Thirteenth National Conference on Artificial Intelligence,* 148-155.

19. Christie, M. and Languénou, E. (2003) A Constraint-Based Approach to Camera Path Planning. In Butz, A., Kruger, A., and Olivier, P., editors, *Smart Graphics, Third International Symposium, SG 2003 Proceedings*, LNCS, Springer, **2733** 172-181.

20. Christie, M., Languénou, E., and Granvilliers, L. (2002) Modeling Camera Control with Constrained Hypertubes. In Hentenryck, P. V., editor, *Principles and Practice of Constraint Programming (CP2002)*, Ithaca, New York, USA, 618–632.

21. Christie, M., Machap, R., Normand, J. M., Olivier, P., and Pickering, J. (2005) Virtual Camera Planning: A Survey. In Butz, A., Fisher, B., Krüger, A., and Olivier, P., editors, *Smart Graphics: Fifth International Symposium Proceedings, SG 2005,* Frauenwörth Cloister, Germany, LNCS, Springer, ISBN 3-540-28179-7, **3638** 40-52.

22. Christie, M. and Normand, J. M. (2005) A Semantic Space Partitioning Approach to Virtual Camera Composition. In Alexa, M. and Marks, J., guest editors, *EUROGRAPHICS 2005,* **24(3)** 247.

23. Dechter, R. (1997) Backtracking Algorithms for Constraint Satisfaction Problems - a Survey. January 29, 1997.

24. Drucker, S. (1994) *Intelligent Camera Control in Graphical Environments.* Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA.

25. Drucker, S. M., Galyean, T., and Zeltzer, D. (1992) CINEMA: A System for Procedural Camera Movements. In Proceedings of *1992 Symposium on Interactive 3D Graphics*, Cambridge, MA, USA, ACM Press, 67-70.

26. Drucker, S. M. and Zeltzer, D. (1994) Intelligent Camera Control in a Virtual Environment. In *Graphics Interface '94*, Banff, Alberta, Canada, 190-199.

27. Drucker, S. M. and Zeltzer, D. (1995) CamDroid: a System for Implementing Intelligent Camera Control. In Proceedings of *the 1995 symposium on Interactive 3D graphics*, Monterey, California, USA, 139-144.

28. Feiner, S. and Seligmann, D. D. (1992) Cutaways and Ghosting: Satisfying Visibility Constraints in Dynamic 3D Illustrations. In *The Visual Computer*, **8(5-6)** 292-302.

29. Freuder, E. C. and Wallace, R. J. (1996) Partial constraint satisfaction. *Artificial Intelligence*, **58(1-3)** 21-70.

30. Giors, J. (2004) The Full Spectrum Warrior camera system. In *Game Developers Conference, 2004.*

31. Gleicher, M. (1992) Briar - a Constraint-based Drawing Program. In *CHI '92 Formal Video Program*, SIGGRAPH video review, 661-662.

32. Gleicher, M. and Witkin, A. (1992) Through-the-Lens Camera Control. In Catmull, E. E., editor, *Computer Graphics (Proceedings of SIGGRAPH '92)*, **26(2)** 331-340.

33. Halper, N. (1999) *Camera Planning for Polyhedral Worlds.* MEng Dissertation, Department of Computer Science, University of York, 1999.

34. Halper, N., Helbing, R., and Strothotte, T. (2001) A Camera Engine for Computer Games: Managing the Trade-off between Constraint Satisfaction and Frame Coherence. In Proceedings of *Eurographics 2001*, **20(3)** 174-183.

35. Halper, N and Olivier, P. (2000) CAMPLAN: A Camera Planning Agent. In Proceedings of *AAAI 2000 Spring Symposium on Smart Graphics*, Stanford, California, USA, 92-100.

36. He, L., Cohen, M. F., and Salesin, D. H. (1996) The Virtual Cinematographer: A Paradigm for Automatic Real-Time Camera Control and Directing. In Proceedings of *SIGGRAPH 96*, New Orleans, LA, USA, 217-224.

37. Hornung, A. (2003) *Autonomous Real-time Camera Agents in Interactive Narratives and games.* Master's thesis, Department of Computer Science V, Aachen University of Technology, Germany.

38. Hornung, A., Lakemeyer, G., and Trogemann, G. (2003) An Autonomous Real-Time Camera Agent for Interactive Narratives and Games. *IVA 2003,* Kloster Irsee, Germany, 236-243.

39. Kyung, M. H., Kim, M., and Hong, S. (1995) Through-the-lens Camera Control with a Simple Jacobian Matrix. In Davis, W. A. and Prusinkiewicz, P., editors, Proceedings of *Graphics Interface '95,* Quebec, 171-178.

40. Jardillier, F. and Languénou, E. (1998) Screen-Space Constraints for Camera Movements: the Virtual Cameraman. In Ferreira, N. and Göbel, M., editors, *Computer Graphics Forum (Proceedings of Eurographics-98),* Blackwell Publishers, ISSN 1067-7055, **17(3)** 175-186.

41. Jermann, C., Trombettoni, G., Neveu, B., and Rueher, M. (2000) A Constraint Programming Approach for Solving Rigid Geometric Systems. In Proceedings of *the Sixth International Conference on Principles and Practice of Constraint Programming (CP2000),* Singapour, LNCS, Springer, 233-248.

42. Jhala, A. and Young, R. M. (2005) Discourse Planning Approach for Cinematic Camera Control for Narratives in Virtual Environments. In Proceedings of *the National Conference of the American Association for Artificial Intelligence,* Pittsburg, PA, USA.

43. Kumar, V. (1992) Algorithms for Constraint Satisfaction Problems: A Survey. *AI Magazine,* **13(1)** 32-44.

44. Languénou, E., Benhamou, F., Goualard, F., and Christie, M. (1998) The Virtual Cameraman: an Interval Constraint Based Approach. In *Constraint Techniques for Artistic Applications (Post ECAI'98 Workshop),* Brighton, UK.

45. LaValle, M., González-Baños, H., Becker, C., and Latombe, J. (1997) Motion Strategies for Maintaining Visibility of a Moving Target. In Proceeding of *IEEE International Conference on Robotics and Automation, ICRA '97,* Albuquerque, New Mexico, USA, **1** 731-736.

46. Li, T. and Xiao, X. (2005) An Interactive Camera Planning System for Automatic Cinematographer. *Eleventh International Conference on Multi Media Modeling (MMM 2005),* Melbourne, Australia, 310-315.

47. Lin, T., Shih, Z., and Tsai, Y. (2004) Cinematic Camera Control in 3D Computer Games. *The Twelfth International Conference in Central Europe on Computer*

Graphics, Visualization and Computer Vision'2004, WSCG 2004, University of West Bohemia, Campus Bory, Plzen-Bory, Czech Republic, 289-296.

48. Mackinlay, J. S., Card, S. and Robertson, G. (1990) Rapid Controlled Movement through a Virtual 3D Workspace. In Proceedings of *ACM SIGGRAPH '90, Computer Graphics,* 24(4) 171-176.

49. Marchand, E., and Courty, N. (2000) Image-based Virtual Camera Motion Strategies. In *Graphics Interface Conference (GI 2000),* 69-76.

50. Marchand, E., and Courty, N. (2002) Controlling a Camera in a Virtual Environment. *The Visual Computer,* **18(1)** 1-19.

51. Montanari, U. (1974) Networks of Constraints: Fundamental Properties and Applications to Picture Processing. *Information Science,* **7(2)** 95-132.

52. Nadel, B. A. (1990) Constraint satisfaction algorithms. *Computational Intelligence,* **5(4)** 188-224.

53. Phillips, C. B., Badler, N. and Granieri, J. (1992) Automatic Viewing Control for 3D Direct Manipulation. In Zeltzer, D., editor, *Computer Graphics (1992 Symposium on Interactive 3D Graphics),* **25** 71-74.

54. Pickering, J. H. (2002) *Intelligent Camera Planning for Computer Graphics.* Ph.D. thesis, Department of Computer Science, University of York, York, YO10 5DD, UK.

55. Platt, J. and Barr, A. (1988) Constraint Methods for Flexible Models. *Computer Graphics (Proceedings SIGGRAPH '88),* **22** 279-288.

56. Ratschan, S. (2002) Approximate Quantified Constraint Solving by Cylindrical Box Decomposition. *Reliable Computing,* **8(1)** 21-42.

57. Turner, R., Balaguer, F., Gobbetti, E., and Thalmann, D. (1991) Physically-based Interactive Camera Motion using 3D Input Devices. In Patrikalakis, N. M., editor, *Scientific Visualization of Physical Phenomena: Proceedings of CG International 1991,* Tokyo, Japan, Springer-Verlag, 135-145.

58. Ware, C. and Osbom, S. (1990) Exploration and Virtual Camera Control in Virtual three-dimensional Environments. In *1990 Symposium on Interactive 3D Graphics,* 175-184.

59. Wei, J. and Li, Z. (1998) On Active Camera Control with Foveate Wavelet Transform. *Technical Report TR 1998-10,* School of Computing Science, Simon Fraser University, Burnaby, BC, Canada.

60. Wilczkowiak, M., Trombettoni, G., Jermann, C., Sturm, P. F., and Boyer, E. (2003) Scene Modeling Based on Constraint System Decomposition Techniques. *ICCV 2003,* 1004-1010.

61. Witkin, A., Fleischer, K, and Barr, A. (1987) Energy Constraints on Parameterized Models. *Computer Graphic*s, **21(4)** 225-232.

Appendix A: ANNOTATION OF SELECTED REFERENCES

1. Amerson, D. and Kime, S. (2000) Real Time Cinematic Camera Control for Interactive Narratives. *American Association for Artificial Intelligence, 2000,* 1-4.

The problem addressed in (Amerson and Kime 2000) is how to apply the cinematographic techniques in the camera control system of the 3D virtual world interactive narratives. The authors begin by referring to the works of (Christianson et al 1996) and state that the camera control concept based on cinematographic constraints of that paper is the foundation of the present work. Then they refer to the works of (He, Cohen, and Salesin 1996) and state that a scene hierarchy to represent cinematography conventions used in that paper is the basis to encode common film idioms into a scene tree. At last they refer to a constraint-based approach called UCAM described in (Bares and Lester 1997) and state that the method of the present paper is a hybrid system based on the three above mentioned methods. The authors propose a real-time camera control system called Film Idiom Language and Model (FILM) that uses cinematographic techniques and inputs from a Narrative Planner to constraint the location and orientation of the camera in a virtual world of the interactive narratives. The film idioms are encoded in scene level. A scene tree is constructed using FILM language. A keyword list is used to search the tree for shot selection. The Narrative Planner generates information based on the actions carried out by the characters and outcome of these actions. This information is translated by a domain specific Translator. A domain independent Director uses the translated scene requirements to select a film idiom from the scene tree. It bounds any unbounded variables in the scene to the virtual world objects. These scene specifications along with the constraints are used by a graphics engine specific Cinematographer to choose the best camera position to satisfy these constraints. If the selected optimal position is not a valid position, constraints are relaxed based on the weights defined in the scene specification using an ad hoc procedural model to find a closest point to the optimal position which satisfy the remaining constraints. The following constraints are used by the method:

1. “lensType”.
2. “lookAt”.
3. “relativeLocation”.
4. “maintainLensType”.
5. “maintainRotation”.
6. “maintainAbsoluteLocation” etc.

The authors explain the camera control architecture in terms of its components: the scene tree, the FILM language, the translator, the director, and the cinematographer. The camera control system was implemented as a part of the Mimesis project. A modified Unreal Tournament (UT) game engine was used as the first development platform. The AI server of the Mimesis system communicates with the modified UT game server. The AI server consists of a HTN-style decomposition planner called Longbow and other supporting subsystems. Longbow formulates plans. The UT server passes user actions to AI server as well as executes the plans formulated by the AI server. The camera control approach was under implementation. The authors claim that after implementation of their

method, it will be able to select abstract film idiom knowledge and convert that knowledge into virtual camera placements.

The authors identify the future work as to use a more general constraint solver.

2. Badros, G. J. (1998) *Constraints in Interactive Graphical Applications.* Ph.D. General Examination, Department of Computer Science and Engineering, University of Washington, Box 352350, Seattle, WA 98195-2350, USA.

The problem addressed in (Badros 1998) is how to make readers familiar with the classes of constraints and solution techniques used in the interactive graphic applications such as drawing, graph layout, visualization, and animation systems. The author summarized performance of backtracking algorithms and suggested several ways to improve their performance in the interactive graphical applications.

The constraints relevant to the geometric applications are identified as: 1) “linear equalities”, 2) “linear inequalities”, 3) “linear geometric”, 4) “geometric”, 5) “geometric (on camera image)”, 6) “geometric in complex plane”, 7) “point-on-object”, 8) “coincident”, 9) “arbitrary acyclic”, 10) “Horizontal relationship between pairs of points (HOR)”, 11) “Vertical relationship between pairs of points (VER)”, 12) “Parallel relationship between pairs of line segments (PARA)”, 13) “Congruence relationship between pairs of line segments (CONG)”, and 14) “Visual Organization Features (VOFs)”.

The constraint solving techniques used are identified as: 1) “relaxation”, 2) “numeric”, 3) “iterative numeric”, 4) “optimized iterative numeric”, 5) “direct numeric (QOCA)”, 6) “differential methods”, 7) “symbolic”, 8) “spring simulation”, 9) “graph-layout, direct & iterative”, 10) “extreme-bound propagation”, 11) “Degree of Freedom (DOF analysis)”, 12) “Constraint Logic Programming with Real arithmetic constraints (CLP(R)-like)”, 13) “Local Propagation (LP)”, and 14) “LP w/o planning”.

The paper is a survey of the classes of constraints and their solution techniques used in interactive graphical applications. The constraints and their solvers used in various systems are shown in table A1. The author classified the constraint solving methods and compared their expressiveness and performance. A visual classification of interactive constraint solvers is given in figure A1 where solvers and sub-solvers are grouped using different kinds of lines, relationships between solvers are shown with the help of different kinds of arrows, and proximity in the figure roughly indicates relatedness. The author observes that the constraints are used to keep relationship among on-screen objects and using constraints is a declarative means for specifying relationship that designers or users wish to hold true, the declarative specification of desired relationship is the fundamental strength of using constraints, and the backtracking algorithm is not successfully used to find solutions in the interactive graphical applications. The author claims that expansion of the classes of the constraints used in an application is a big challenge because the classes of constraints used by the application are restricted by its constraint solver.

The author identifies the following future work:

- To develop fast, expressive and understandable reusable constraint solvers.
- To reuse constraint solvers.
- To make debugging of constraints easier for the users.

	System	Author (Year)	Constraints supported	Solving technique	Performance
Drawing	Sketchpad	Sutherland (1963)	geometric	LP, relaxation	$O(n)$, $O(n^2)$
	IDEAL	Van Wyk (1982)	geometric in complex plane	LP w/o planning	$O(n^2)$
	Juno	Nelson (1985)	CONG, PARA, HOR, VER	iterative numeric	$O(n^3)$
	Juno-2	Heydon and Nelson (1994)	CONG, PARA, HOR, VER	optimized iter. num.	$O(n^3)$
	Briar	Gleicher and Witkin (1994)	points-on-object,coincident	differential methods	$O(n^3)$
	Unidraw	Helm et al. (1995)	linear (in)equalities	direct numeric (QOCA)	$O(n^3)$, $O(n^2)$
	GCE	Kramer (1992)	geometric	DOF analysis	$O(ng)$, $O(n \log n)$
	Chimera	Kurlander (1991)	geometric	symbolic, numeric	$O(n^2)$
	Pegasus	Igarashi et al. (1997)	geometric	CLP(R)-like	$O(2^n)$
Graph	GLIDE	Ryall et al. (1997)	VOFs	spring simulation	polynomial
	CGL	He et al. (1996)	linear (in)equalities	iter. numeric	>1 sec (tree $n = 16$)
Visualization	TRIP*N*, IMAGE	Takahashi et al. (1998)	linear geometric	graph-layout, direct & iterative	"needs to be faster"
	ICOLA	Oster & Kusalik (1998)	linear inequalities	extreme-bound propagation	$O(n + v)$
	Penguins	Chok & Marriott (1998)	linear (in)equalities	direct numeric (QOCA)	interactive ($n \leq 700$)
Animation	TLCC	Gleicher & Witkin (1992)	geometric (on camera image)	differential methods	$O(n^3)$, non-interactive
	Animus	Duisberg (1987)	arbitrary acyclic	LP, relaxation	$O(n)$, $O(n^2)$
	JIM, Parcon	Griebel et al. (1996)	linear (in)equalities, geometric	iterative numeric	<1 sec ($n \leq 100$)

Table A1: List of constraints and solvers in interactive graphical applications (Badros 1998; page 3; table 1)

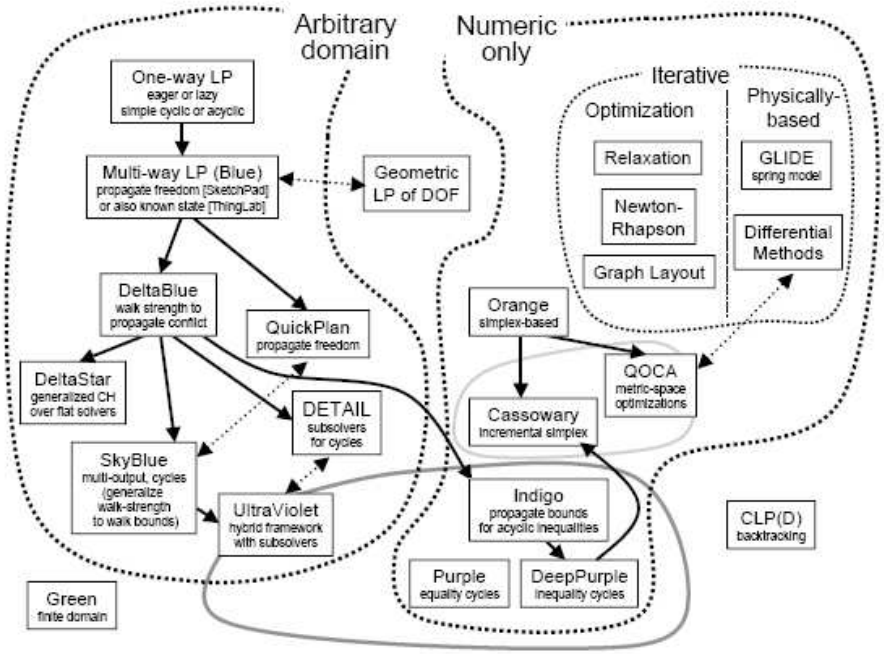

Figure A1: Classification of Interactive Constraint Solvers (Badros 1998; page 10; figure 1)

3. Bares, W. H. and Lester, J. C. (1999) Intelligent Multi-shot Visualization Interfaces for Dynamic 3D Worlds. In *IUI'99: Proceedings of 1999 International Conference on Intelligent User Interfaces,* Los Angeles, California, USA, 119-126.

The problem addressed in (Bares and Lester 1999) is how to enhance abilities of the visualization interface of 3D virtual environment to create real-time dynamic visualizations that effectively respond to viewing request of the users. The authors propose a real-time camera visualization interface system for dynamic 3D world called CONSTRAINTCAM. The proposed intelligent visualization interface uses a partial constraint based framework. The main modules of the system are the Constraint Analyzer, Constraint Solver, and Multi-Shot Frame Composer. Visualization goal of a user is expressed as a constraint problem. Viewers can select which subject(s) to be viewed, specify constraints for each subject(s), select cinematic style or pace etc. Given the viewing goal, the Constraint analyzer will determine a consistent region where all the constraints will be satisfied. The Constraint Solver will search the consistent region to find a camera placement that will satisfy all the constraints. If such a solution is not possible, the Constraint Solver relaxes pairs of weakest incompatible constraints in steps to find a solution. If a solution is not found even after constraint relaxation, the problem is decomposed to find multi-shot solution. In that case, the Multi-Shot Frame Composer creates a sequential or composite visualization solution. The method supports the following four constraints on the subject(s) to be viewed:

1. "Vantage Angle".
2. "Viewing Distance".
3. "Occlusion Avoidance".
4. "Room Enclosure".

The authors describe the system with an example interaction in an interactive 3D testbed. The example interaction consists of multiple autonomous characters interacting in a dense cityscape. The test environment was a Pentium II 333MHz processor with Permidia2 OpenGL accelerator. The authors report that the frame rate achieved by the method was approximately 7 to 15 frames per second. The real-time performance of the method and the results of an informal focus group study with viewers are reported as encouraging. The authors claim that by using constraints relaxation and creating multi-shot solutions, the system can compute near optimal solution to difficult problems.

The authors identify the following future work:

- To add a direct manipulating interface to select subject(s) and values of any of the constraints.
- To use the silhouettes instead of bounding boxes of irregular and concave polyhedral occluding objects to handle occlusion avoidance.
- To develop a more complete representation of distance of potential occluding objects.

The results of the paper have been referred to by (Bares, Thainimit, and McDermott 2000).

4. Bares, W., McDermott, S., Boudreaux, C., and Thainimit, S. (2000) Virtual 3D Camera Composition from Frame Constraints. In *Eight ACM International Conference on Multimedia*, Marina Del Ray, California, USA, 177-186.

The problem addressed in (Bares et al 2000) is how to define camera compositions efficiently avoiding both pre-defined camera placements and script generated constraints in 3D environments. The authors propose a system consisting of a graphical interface called the Storyboard Frame Editor and a recursive heuristic constraint solver. The frames drawn on the editor are automatically encoded into a set of constraints. The editor export files in two forms of shot definition namely the relative displacement definition and the constraint definition. The relative displacement definition is used by the solver for an instantaneous solution if configuration of scene objects match with those of the editor's 3D scene model. Otherwise, the solver use exhaustive generation and test method. At first it finds a valid region for each constraint. Then it examines candidate camera shots by discretely incrementing camera parameters inside the respective valid regions. Each camera placement is evaluated with the help of individual constraint evaluation function to find individual constraint satisfaction rating. The cumulative constraint satisfaction rating is the weighted sum of the individual constraint satisfaction ratings. The search begins at relatively coarse resolution candidate shots to find pre-defined numbers of best candidate shots based on cumulative satisfaction ratings. The search space around the best candidate shots are recursively searched with increasingly refined resolution. If no camera position is found which satisfies all the constraints, the solver reports failure.

Fifteen constraints are supported by the method including the following constraints:

1. "CAM_POS_IN_REGION".
2. "OBJ_EXCLUDE_OR_HIDE".
3. "OBJ_IN_FIELD_OF_VIEW".
4. "OBJ_OCCLUSION_MINIMIZE".
5. "OBJ_OCCLUSION_PARTIAL".
6. "OBJ_PROJECTION_ABSOLUTE".
7. "OBJ_PROJECTION_SIZE".
8. "OBJ_VIEW_ANGLE" etc.

The authors describe functionality of the Storyboard Frame Editor, important constraints listed above, and the heuristic constraint solver algorithm. The constraints used in the paper are similar to the constraints used in (Bares, Thainimit, and McDermott 2000). The test environment was an Intergraph GL2 computer with Pentium II 400MHz processor, 256 MB RAM, and VX113 AGP OpenGL accelerator. Four camera shot problems are explained and their solution shots are shown in the figures 10-14 of the paper. The cumulative constraint satisfaction rating of the four shots are shown in the table A4 based on two implementations of the constraint solver. The authors claim that the Storyboard Frame Editor and the constraint solver together is a tool to define how the virtual camera should film objects of interest without worrying about the placement of the camera.

Figure	*Heuristic Search*	*Relative Displacement*
10	0.877109	0.0
11	0.850244	0.338701
12	0.774565	0.338701
14	0.810209	0.473219

Table A4: Cumulative Constraint Satisfaction Rating for Four Solution Shots given in (Bares et al 2000; page 186; table 3)

The authors identify the following future work:

- To optimize the search heuristic and constraint evaluators to obtained real-time performance.
- The constraint solver module is to be added with several prototypes of interactive 3D virtual environments.
- To add additional constraint types by coding new evaluators and script parsing functions.
- To add capability to handle failure to find a constraint solution, either less important constraint will be relaxed or multiple shots visualization will be used.
- To analyze effectiveness of the user interface and aesthetics of the computed shots.

5. Bares, W. H., Thainimit, S., and McDermott, S. (2000) A Model for Constraint-based Camera Planning. In *AAAI 2000 Spring Symposium Series on Smart Graphics*, Stanford, California, USA, 84-91.

The problem addressed in (Bares, Thainimit, and McDermott 2000) is how to plan camera shots automatically in virtual 3D environments such that each shot will communicate some specific visual message or goal. The authors begin by referring to their previous work in (Bares, Grégoire, and Lester 1998) and (Bares and Lester 1999) and state that those work served as a starting point for this paper. The authors propose a constraint based camera planning method for 3D virtual environments. A user or a software system requests to visualize subjects of interest and how they should be viewed. The view request will be translated by constraint script into a set of constraints. The constraint solver tries to find a solution camera shot based on given constraints. It uses an exhaustive generate-and-test method to determine the camera placement having the highest cumulative constraint satisfaction rating. The cumulative constraint satisfaction rating is weighted sum of all individual constraint satisfaction measured as product of relative priority and satisfaction rating of the constraint.

Fifteen constraints are supported by the method including the following constraints:

1. “CAM_FIELD_OF_VIEW”.
2. “CAM_POS_IN_REGION”.
3. “LOOK_AT_POINT”.
4. “OBJ_DEPTH_ORDER”.
5. “OBJ_DISTANCE”.
6. “OBJ_IN_FIELD_OF_VIEW”.
7. “OBJ_OCCLUSION_MINIMIZE”.
8. “OBJ_PROJECTION_ABSOLUTE”.

9. “OBJ_PROJECTION_RELATIVE”.
10. “OBJ_PROJECTION_SIZE”.
11. “OBJ_VIEW_ANGLE” etc.

The authors describe the method and define important constraints listed above. The constraints used in the paper are similar to the constraints used in (Bares et al 2000). An example application of the method to find a camera position for an over the shoulder shot during a conversation between two players is explained. The test environment was a Pentium II 400MHz processor. The constraint solver searched camera position over a 20x20x20 grid. The aim direction was at 15° interval and field of view angle was over a range of 10 values. The authors report that a total of 13,891,500 shots were evaluated in approximately 30 minutes to determine the solution shot with 89% satisfaction rating. The virtual camera constraint model described in the paper was under development. The authors claim that the constraints can be used to create cinematic camera shots and to communicate specific visual message.

The authors identify the following future work:

- To develop a more efficient constraint solution search algorithm.
- To develop a WYSIWYG graphic interface to generate automatically constraint script for a desired camera shot.

The results of the paper have been referred to by (Lin, Shih, and Tsai 2004).

6. Benhamou, F., Goualard, F., Languénou, E., and Christie, M. (2004) Interval Constraint Solving for Camera Control and Motion Planning. *ACM Transactions on Computational Logic*, **5(4)** 732-767.

The problem addressed in (Benhamou et al 2004) is how to find an efficient solution of control and motion planning problems and enable a graphic artist to specify the desired camera movements for a shot using cinematographic idioms. The authors observe that many control and motion planning problems can be reduced. The reduced problems become either to find a sound approximate solution space determined by a set of nonlinear equalities or the guaranteed tuning problem. In the guaranteed tuning problem reduction method, a value for some tuning parameter is found in order to verify a set of inequalities for all possible values of some perturbation vector. The classical approaches to solve these problems are based on quantifier elimination techniques. The constraint solver used for the present work utilized interval arithmetic and local consistency enforcement based sound numerical methods to prune the search space. Each constraint is considered one by one along with the sets of elements verifying all the constraints considered so far.

The authors present and prove soundness for the inner contracting operator based inner propagation algorithm (IPA) and two inner contracting operator algorithms (ICO1 and ICO2). The method IPABC is the algorithm IPA using ICO2 which in turn uses ‘Outer-Box Contracting Operator’. They also present, analyze, and prove various concepts and operators relevant to the algorithms. They describe interval constraint solving using local

consistencies. They investigate application of their method to camera control problems. The user defined descriptions for a shot are translated into constraints in terms of camera parameters. Constraints used in the paper are similar to the constraints used in (Jardillier and Languénou 1998) and (Languénou et al 1998). The authors state to have conducted experiments with the help of a prototype of a declarative modeler which used cinematic languages to specify camera motion. The prototype was implemented in C++ and Tcl/Tk. The test environment was a Pentium III, 800 MHz Linux PC. To generate all solutions for the nine benchmark problems described in the paper, IPABC method was 6.2 to 283 times faster than JLA method of (Jardillier and Languénou 1998). The authors observe that the efficiency of their method increased steadily with progressively higher precision. They claim that their method is much faster than both the soundly solving system of nonlinear inequalities and the guaranteed tuning problem with one dimensional perturbation vector.

The authors identify the following future work:

- To speed up interval constraint-based solvers.
- To allow camera movement along all the degrees of freedom with reduced number of variables.
- To compare inner approximation of relations computation using "Kaucher arithmetic" and using outer contracting operators.

7. Bourne, O. and Sattar, A. (2005) Applying Constraint Weighting to Autonomous Camera Control. In *Artificial Intelligence and Interactive Digital Entertainment*, Marina Del Ray, CA, USA, AAAI Press, 3-8.

The problem addressed in (Bourne and Sattar 2005) is how to implement autonomous camera control for third-person perspective computer games with manageable implementation complexity and system usability. The authors begin by referring to their previous approach of (Bourne and Sattar 2004) and state that this paper extends their previous work. The authors applied constraint weighting techniques in the paper. Each of the constraints has two parameters. One is for the desired value and another is for the weight of the constraint. The weights will indicate preferences among infeasible constraints. The weights are unique and normalized. Different kind of camera behaviours can be achieved by manipulating the weights. The cost of a constraint is the weighted difference between its desired value and current potential solution value. The total cost of a solution for a camera position in a frame is the sum of the costs for individual constraints except 'frame coherence' constraint. An incomplete Stochastic Local Search technique is applied to find the lowest cost solution for the constraints for a frame. The search method makes a pre-defined numbers of moves and returns the best solution found so far as optimal solution. The 'frame coherence' constraint prevents camera movement if cost improvement does not exceed the constraint's value and act like a simple acceleration and deceleration mechanism. A meta-level visibility constraint using ray casing from the target to the camera is applied to influence the underlying constraint solver for occlusion avoidance. The proposed system supports the following four constraints:

1. "Height".

2. “Distance”.
3. “Orientation”.
4. “Frame Coherence”.

The camera system was implemented within a 3D game engine with dynamic environment support with changing or multiple target objects. The test environment was an AMD Athlon 2800+ processor, 512MB RAM, Windows XP Service Pack 2 PC. The test 3D world has around 225,000 triangles and 350 bounding volumes where a combination of scripted and interactive control of the target object is used. The authors state to have conducted an experiment and report that the camera system takes 0.00094 seconds per frame on average to perform both the visibility test using ray-casting and constraint solving. The authors claim to have demonstrated that autonomous camera control can be significantly simplified, and the gap between implementation and usability requirements can also be significantly reduced by their technique.

The authors identify the following future work:

- To develop a more efficient heuristics for local search algorithm.
- To extend the viewing properties of the camera with new constraints.
- To generate constraint weights by using a genetic algorithm so that the interactive camera can match a scripted camera movement.
- To investigate application of the camera system into more sophisticated autonomous cinematography system.

8. Christie, M. and Languénou, E. (2003) A Constraint-Based Approach to Camera Path Planning. In Butz, A., Kruger, A., and Olivier, P., editors, *Smart Graphics, Third International Symposium, SG 2003 Proceedings*, LNCS, Springer, **2733** 172-181.

The problem addressed in (Christie and Languénou 2003) is how to determine camera path according to user given declarative properties such as locations or orientations of objects on the screen at a given time. The authors begin by referring to the approach of (Jardillier and Languénou 1998) and state that they have followed that approach. The authors propose a constraint based camera control method to find camera paths based on cinematographic techniques. The paths are based on user defined sequences of parameterized elementary camera movements called hypertubes. The search algorithm combines constraint propagation and enumeration to compute a solution of the global NCSP consists of individual hypertube NCSPs. The solver uses interval-based filtering to remove areas having no solutions for a hypertube and backtracking to remove inconsistent hypertubes.

The authors describe the hypertube model and the search algorithm. A C++ based prototype was used for implementation of the model. The test environment was a Pentium III, 800MHz Linux PC. The authors state to have conducted experiments. The results are shown in the table A6 for two examples. The scenes for the test consist of mobile objects A and B, and a static object C. In addition, the scene of example #2 has obstacles shown as red circles. Camera paths are calculated for a traveling, a local arching, and panoramic camera movements. The paths are shown in top and bottom part

of figure A3 for Example #1 and Example #2 respectively. The authors observe that time required by the approach of the paper to generate camera paths is around 3 to 6 seconds for problems defined with 26 variables in comparison to the time needed by the earlier approach of (Jardillier and Languénou 1998) is more than 10 minutes for the problem defined with more than 7 variables only (both for the first solution). The test results are reported as very encouraging and as a consequence open up the way for real-time application of the approach. The authors claim that the approach fine tunes the images in the shots following cinematographic idioms and also overcomes expensive computations, restricted movements, and poor interactivity problems of the approach of (Jardillier and Languénou 1998).

Problem	Hypertubes	Variables	Constraints	First solution (ms)
Example #1	3	26	17	3170
Example #2	3	26	25	6090

Table A6: Computation Time for Two Problems (Christie and Languénou 2003; page 9; table 4)

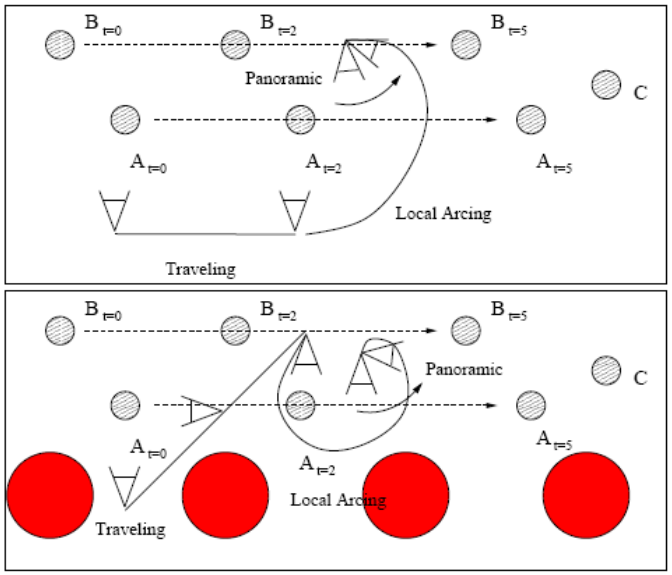

Figure A3: Calculated Camera Path (Christie and Languénou 2003; page 9; Fig. 4)

The authors identify future work as to generation of the hypertubes on the fly based on the events generated in the 3D scene.

9. Christie, M., Languénou, E., and Granvilliers, L. (2002) Modeling Camera Control with Constrained Hypertubes. In Hentenryck, P. V., editor, *Principles and Practice of Constraint Programming (CP2002)*, Ithaca, New York, USA, 618–632.

The problem addressed in (Christie, Languénou, and Granvilliers 2002) is how to determine a camera path according to user given declarative properties such as locations or orientations of objects on the screen at a given time. The authors begin by referring to the approach of (Jardillier and Languénou 1998) and state that they have followed that

approach. The authors propose a constraint based camera control method to find camera paths based on cinematographic techniques. The paths are based on user defined sequences of parameterized elementary camera movements. These cinematic primitives are called hypertubes. The hypertubes are defined for linear movement (traveling), rotation around the camera's horizontal or vertical axes (panoramic), zoom-in, zoom-out, and arcing to turn around objects. The hypertubes are connected by intertubes which are relations between two successive hypertubes that guarantee continuity of camera movement. A hypertube with a set of properties and duration is a Numeric CSP (NCSP). Each cinematographic property is redefined as constraints on the hypertube variables. To model a shot, a user like a director of a film provides properties like orientations or locations of the objects on the screen and a sequence of elementary camera movements. Then interval consistency and quantified constraint based solving algorithms will compute parameters of the hypertubes. The search algorithm combines constraint propagation and enumeration to compute a solution of the global NCSP consists of individual hypertube NCSPs. The solver uses interval-based filtering to remove inconsistencies and isolate inner boxes for each hypertubes. It finds a solution based on optimization criterion from the set of the isolated inner boxes of the given hypertube in order to determine starting conditions for the next hypertube. It uses Tabu strategy based backtracking to remove inconsistent hypertubes.

The authors describe the hypertube model and the search algorithm. The OpAC C++ library with an extension to manage universally quantified constraints is used to implement the constraint solving algorithm. The test environment was a Pentium II, 350MHz Linux PC. The authors state to have conducted experiments. The numerical results obtained are shown in the table A5 for the three examples. The calculated camera path for Example #2 of the table A5 is shown in the figure A2. The 3D scene for Example #2 has a static object A and a moving object B. The shot consists of three basic movements: a horizontal panoramic from A to B, an arching around B, and another horizontal panoramic to follow B. The execution time reported for Example #2 is around 5 seconds for a 6 seconds animation. The authors observe that time required by the approach of the paper to generate camera paths is around 4 to 10 seconds for the problems defined with 18 to 48 variables in comparison to the time needed by the earlier approach of (Jardillier and Languénou 1998) is more than 10 minutes for the problem defined with more than 7 variables only (both for the first solution). The authors claim that their approach shows a great improvement in time and animation quality over the former approaches and the encouraging experimental test results open exciting perspectives to apply it for visual occlusion handling etc.

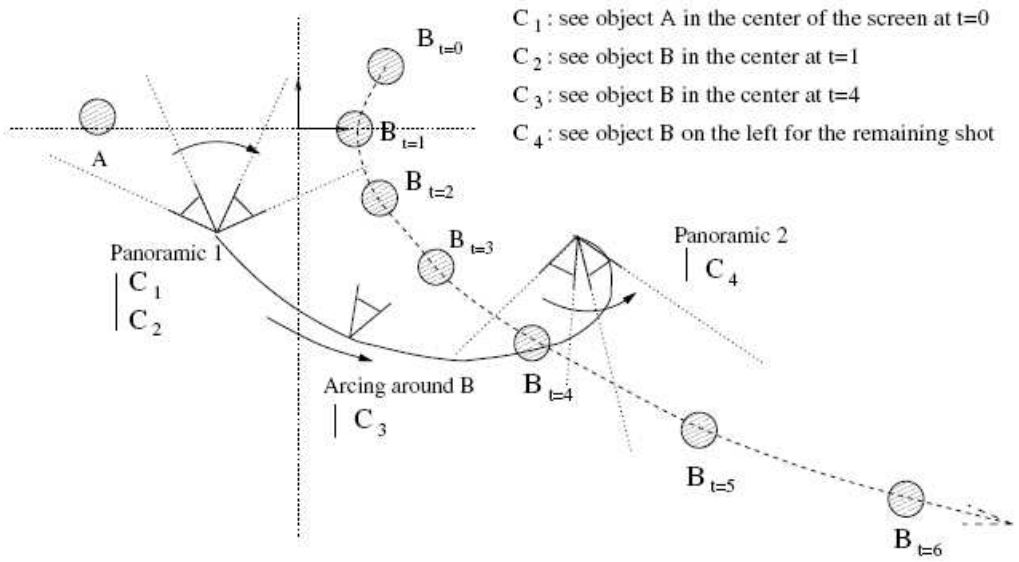

Figure A2: Calculated Camera Path for Example #2 (Christie, Languénou, and Granvilliers, 2002, page 629, Fig. 11)

Problem	Hypertubes	Variables	Constraints	First solution (ms)
Example #1	2	18	9	3940
Example #2	3	22	12	5290
Example #3	6	48	24	9880

Table A5: Computation Time for Three Problems (Christie, Languénou, and Granvilliers, 2002, page 629, table 2)

The authors identify the following future work:

- To find the closest solution to the user modification of the location or orientation of objects in the 3D space.
- To generate the hypertubes on the fly considering events in the 3D scene.
- To search and implement other constraint satisfaction problems expressed by means of CSP decomposition and variable linking.

10. Christie, M., Machap, R., Normand, J. M., Olivier, P., and Pickering, J. (2005) Virtual Camera Planning: A Survey. In Butz, A., Fisher, B., Krüger, A., and Olivier, P., editors, *Smart Graphics: Fifth International Symposium Proceedings, SG 2005,* Frauenwörth Cloister, Germany, LNCS, Springer, ISBN 3-540-28179-7, **3638** 40-52.

The problem addressed in (Christie et al 2005) is how to make automated virtual camera planning techniques more understandable to the readers. The authors have presented in the paper an overview of automated camera planning techniques and structured the survey on the expressivity of the set of properties and the characteristics of the solving mechanism. Three levels of image properties are proposed. They are geometric, perceptual, and aesthetic. However most of the actual systems rely solely on the geometric properties.

The paper is a survey of camera control systems. At the beginning, the requirements of camera planning systems based on shot properties are discussed. The camera control systems are classified as first person, third person, and action replay camera systems. Then the camera planning techniques around expressivity and solution mechanisms are discussed. The techniques based on on-camera and on-screen properties of expressivity are described. The on-camera methods include both static and dynamic systems. Four types of existing camera planning approaches based on solving mechanisms are described as algebraic, interactive, reactive real time, and optimization and constraint-based systems. The optimization and constraint-based systems are classified as complete, incomplete, and hybrid approaches. The complete method includes hard constraint CSP and hierarchical constraint methods. The incomplete method includes constraint-optimization and pure optimization methods. The authors observe that solution technique adopted by the approaches constrains both geometric abstraction and expressiveness. The authors claim that they have identified and described in the paper the principle shortcomings of the existing techniques of automated camera planning.

The authors identify the following future work:

- To improve quality of abstraction of 3D objects with an adequate model as well as an effective solving mechanism.
- To develop a model for constraint based systems to decide weight of each value and aggregation of the weights in a single objective function.
- To develop a cognitive model which will incorporate perceptual and aesthetic properties to assist users to construct time and space characteristics of the virtual world in their mind.

11. Drucker, S. M. and Zeltzer, D. (1994) Intelligent Camera Control in a Virtual Environment. In *Graphics Interface '94*, Banff, Alberta, Canada, 190-199.

The problem addressed in (Drucker and Zeltzer 1994) is how to implement intelligent camera controls in a 3D virtual environment. The authors start by referring to their previous work in (Drucker, Galyean, and Zeltzer 1992) on procedural framework for camera control and state that problems in devising a general procedure for camera control led them to formalize the constraint-based framework. They also state that the paper is based on camera specifications used in the computer graphics and path finding methods used in the robotics. The authors present a design methodology for a framework to control the camera in a 3D virtual environment. The authors observe that a camera is controlled for particular reasons which can be represented as constraints on the camera parameters. The constraints can be identified from analysis of task requirements. Thus at first, task requirements for a specific environment are to be analyzed. Next, camera modules are built with constraints identified from the task requirements. The concept of camera module of the paper is analogous to the concept of shot in the cinematography. Finally, a group of user interfaces are to be connected to the framework. The following are examples of constraints used in the method:

1. "maintain the camera's up vector to align with world up".
2. "maintain height relative to the floor".

3. “maintain the camera’s gaze (i.e., view normal) toward a specified object”.
4. “maintain the camera’s position on a collision free path through world”.

The authors implemented a camera control framework for a virtual museum. The Virtual Museum system consists of a general interpreter, a built-in renderer, an object database, and a network of camera modules along with user interfaces. The authors identified tasks to be performed in the virtual museum. They describe the camera modules which will encapsulate the tasks as constraints. A camera module consists of an initializer, a controller, a list of constraints, and state components. A constraint solver finds final camera parameters for a camera module satisfying all the constraints of the module. Active camera module handles all of the user inputs. The authors describe the path planning camera module to navigate through the museum. It is identified as the most complicated module. Path planning is done in steps. The A* algorithm is used to find which rooms are to be visited. The path through a room is found using a numerical navigation function. Tour plans though the virtual museum is also described. The interpreter and user interface were implemented in TCL and TK respectively. The test environment was a W3D system. The W3D system is an extension to the 3D virtual environment software testbed of MIT. The authors claim that using their method special constraints can be designed and combined with other constraints for camera control.

The authors identify the following future work:

- To add more efficient path planning algorithm to deal with totally dynamic environment.
- To add variety of constraints from cinematography into the camera framework.

The results of the paper have been referred to by (Lin, Shih, and Tsai 2004).

12. Giors, J. (2004) The Full Spectrum Warrior camera system. In *Game Developers Conference, 2004.*

The problem addressed in (Giors 2004) is how to implement an effective camera control system in commercial computer games. The author observes that the basic Proportional Controller (PC) is a common camera motion control system in commercial computer games. PC has nice arrival characteristics. The disadvantages of PC are abrupt exit characteristics and lag of current camera position behind the moving targets. To solve the problems of PC, Modified Proportional Controller (MPC) is used to control camera motion in FSW game. MPC controls exit characteristics by limiting acceleration and lag is corrected considering velocity of the target points.

The author describes the camera system of FSW. FSW has several cameras. Each camera is derived from an abstract base class. The cameras managed by a camera manager may operate simultaneously or independently. The main camera represents the primary user controlled view and the fly-by system. The other cameras are for cinematic, in-game events, and playback. The cinematic camera can move between preset positions in addition to the movements of normal in-game view system. The camera system remains under control of the user all the time except at the beginning of a fly-by. FSW has an

auto-look feature to look around corners. This feature will handle occlusion using ray-casting from the camera position. On the other hand, FSW uses ray-casting from the 'lookat' object to avoid collision during normal view and fly-by action. FSW uses camera cuts to avoid passing through obstacles and whenever the destination is beyond a preset distance. The author observes that multiple independent cameras are more useful due to different viewing requirements and one general purpose camera that does everything is likely to lead to a highly complex system. The FSW camera system is affected by over 100 parameters which pose a tuning problem for the designers or programmers. The author claims that unique solutions to several camera control problems were found for FSW due to special attention being given to the issues related to the implementation of camera system during development of the game.

The author identifies the following future work:

- To re-implement the main camera with several cameras to simplify tuning problem.
- To integrate traditional cinematic camera ideas from the film industry.
- To replace linear ray-hits by volumetric examination to improve the operations of collision avoidance, auto-look etc.
- To update feedback system during tuning process.

13. Halper, N., Helbing, R., and Strothotte, T. (2001) A Camera Engine for Computer Games: Managing the Trade-off between Constraint Satisfaction and Frame Coherence. In Proceedings of *Eurographics 2001,* **20(3)** 174-183.

The problem addressed in (Halper, Helbing, and Strothotte 2001) is how to implement an automatic and dynamic camera control system in computer games which obtains a balance between optimality of camera setting and smooth transition with appropriate cuts. The camera module of the method consists of a Director and a Predictive Camera Planner. Based on past trajectory and acceleration, scene and object states at a future time are calculated by the Planner. The current trajectory is modified by the Planner so that the camera will be at the calculated position at that time. It achieves frame coherence by computing a new camera state using existing camera states. The camera position for the next frame is estimated along the path. Then it requests the Director for a setting for the future time. The events created by the action module of the game pipeline are used by the Director to select templates from its Shot Library with the help of its Template Selector. To fit successive shots together, various transition rules are utilized to select templates. The Director selects constraints using the templates from its Shot Template for the Constraint Solver of the Planner and uses its Emotion Template to influence the result of camera shots. Then the Constraint Solver is applied by the Planner on the estimated position to find the camera state for the next frame. The Constraint Solver finds a solution for the constraints in hierarchal order in such a way that a successive constraint solution influences previous solutions minimally. Each constraint has an optimal setting, a tolerance region, and a relaxation parameter. The relaxation parameter will determine how far the camera can be placed from the optimal value. The camera is placed at the boarder of the tolerance region of a constraint if it lies outside of the region. The occlusion avoidance is handled automatically by generating a potentially visible region

using coloured polygons and an image buffer. The brightness is used to indicate the preferred region to avoid occlusion. The system supports the following seven constraints:

1. “Height angle”.
2. “Level at”.
3. “Facing”.
4. “Angle to line of interest”.
5. “Size”.
6. “Visibility”.
7. “Look at”.

The authors set up experiments to explore virtual environments. An exploration template was implemented using four constraints namely ‘level at’, ‘size’, ‘visibility’, and ‘look at’. With this constraint set, the camera should follow the character at its height and keep it in view without obstruction all the time. Three scenarios namely a helicopter flying through a city, a human exploring inside of a medieval building, and a bee flying through a highly cluttered attic were explored using the template. The authors report that the camera performed remarkably well in all the cases as it avoided obstructions and collisions, and was able to adjust appropriately. They claim that their camera system is the first frame-coherent constraint-based camera engine and produces “nearest-best-fit” frame coherent camera solution in real time by considering future conditions.

The authors identify the following future work:

- To solve for various visual goals based on the coherent results produced by the method.
- To develop new types of camera management by adding and learning new techniques using evolutionary design to achieve more sophisticated navigational goals and add more predictive techniques.

The results of the paper have been referred to by (Lin, Shih, and Tsai 2004).

14. Hornung, A., Lakemeyer, G., and Trogemann, G. (2003) An Autonomous Real-Time Camera Agent for Interactive Narratives and Games. *IVA 2003*, Kloster Irsee, Germany, 236-243.

The problem addressed in (Hornung, Lakemeyer, and Trogemann 2003) is how to transfer cinematographic knowledge of camera control to interactive narratives and games. The authors formalize the dramaturgical means of expression of movie camera systems to be applied to interactive narratives. They identify a set of basic dramaturgical principles and use eight parameters of events to communicate information about states of a story between a narrative application and a camera agent. A story is represented within the camera agent as narrative events in such a way that a narrative event corresponds to a single story event. The narrative application will send information regarding its states as narrative events to the camera agent. After receiving an event, the camera agent re-computes its internal representation of the narrative based on the events received so far. Then the camera agent selects an event as the active one for visualization based on the history of the narrative, coherence between the events, event priority etc. Next, the

decision module of the camera agent selects matching shots from user-defined shot library to be added to a priority list of the active event. Finally, the action-realization module of the camera agent computes the camera position, orientation etc. using a cinematic rule for the shot with highest priority for the active event and transfers the information back to the narrative application for visualization. If a shot can not be realized, the next matching shot will be considered.

The authors developed a real time autonomous camera agent. To test the camera agent, the authors modified an existing computer game called Half-Life. Modified Half-Life will generate a narrative event whenever a character changes its internal state. The authors claim to have conducted an experimental evaluation by letting professional cinematographer and non professional audiences view both the original first-person view of the game and the shots created by the camera agent. The result of the experiment is reported as "very convincing and significantly enhanced the narrative experience of Half-Life for the spectator". The authors claim that their system, as a part of an interactive narrative environment, demonstrated the practicality of the system.

The authors identify the following future work:

- To understand and formalize more complex cinematic concepts.
- To integrate planning techniques to allow sophisticated reasoning about visual outcome of camera shots such as geometric constraints within a scene to support the narrative.

15. Jardillier, F. and Languénou, E. (1998) Screen-Space Constraints for Camera Movements: the Virtual Cameraman. In Ferreira, N. and Göbel, M., editors, *Computer Graphics Forum (Proceedings of Eurographics-98)*, Blackwell Publishers, ISSN 1067-7055, **17(3)** 175-186.

The problem addressed in (Jardillier and Languénou 1998) is how to help graphic designers create camera movements at the same time avoiding all the technical details. They begin by referring to the 'Declarative modelling' approach of the geometric modeling community with three phases namely description, generation, and result exploration and state that the present work follows that approach. A user of the method can define constraints either on the image space or on the object of the scene or on both.

Constraints on the camera are:

1. "fixed location panoramic shot".
2. "pure traveling".
3. "high angle".
4. "low angle" etc.

Constraints on the image space are:

1. "fully included in".
2. "partially included in".
3. "fully excluded of" etc.

Constraints on the object of the scene are:

1. “orientation of the object axis”.
2. “upside down”.
3. “stand up”.
4. “front side visible”.
5. “closer than” etc.

Time is also considered as a variable. Thus constraints can be defined for any arbitrary duration of animation. A global solving process computes a solution set of camera movements which satisfies multiple constraints for the complete animation. The constraint solver is based on interval arithmetic. It recursively evaluate the current interval function on hyper-rectangles using tri-valued logic. If evaluating response for a hyper-rectangle is true, it is a solution and will be added to the solution set. If evaluating response for a hyper-rectangle is false, sub hyper-rectangles of the hyper-rectangle will not be evaluated further. If evaluating response for a hyper-rectangle is unknown, the hyper-rectangle is subdivided into two sub hyper-rectangles which will be evaluated further similarly.

The authors describe mathematical models for the scene objects, camera, and their movements. Then constraints, constraint solver and three ways to include time variable namely ‘brute force’, ‘width first’, and ‘deep first’ methods are described. The authors state to have conducted an experiment. The test environment was a Dec Alpha 250 computer. The time required to obtain a solution set for a static and a dynamic camera problem was reported as 1 hour 30 minutes and 10 hours respectively. The authors claim that it is the first method that can compute a complete set of camera positioning solutions, the user given constraints are guaranteed to be satisfied because no key framing or interpolation is used, and need no external solver to implement the method.

The authors identify the following future work:

- To provide a way to present solution sets and navigate through it intelligently.
- To obtain solution set in near real time.
- To examine if visibility testing is required for the method instead of the Z-testing currently employed.

The results of the paper have been referred to by (Languénou et al 1998), (Christie, Languénou, and Granvilliers 2002), (Christie and Languénou 2003), and (Benhamou et al 2004).

16. Jhala, A. and Young, R. M. (2005) Discourse Planning Approach for Cinematic Camera Control for Narratives in Virtual Environments. In Proceedings of *the National Conference of the American Association for Artificial Intelligence,* Pittsburg, PA, USA.

The problem addressed in (Jhala and Young 2005) is how to communicate effective information to the users of complex narrative-based virtual environments using camera control. The authors propose a camera planning system for narratives in virtual environments where decisions are made in three levels: selection of cinematic geometric

composition, camera parameter for information communication, and camera shots and transition to maintain rhetoric coherence. The camera planning system is similar to the film production pipeline. The film idioms are formalized to communicate effective information and represented as plan operators. The representation of idioms has casual motivation and hierarchy. It allows the system designer to rank candidate shot sequences. The story is represented as a sequence of actions executed in the 3D world. The knowledge base of the discourse camera planner consists of the sequence of action executed in the 3D world, and the annotations indicating properties and characters of the story. The planning system incrementally formulates plans by adding steps to the existing plan to satisfy a goal state or a pre-condition of an existing action by the effect of the action, or to refine an abstract action by expansion. It uses causal reasoning, temporal reasoning, and decomposition during addition of steps. Then a ranking function with user preference chooses the best plan. The discourse camera planner utilizes a library of hierarchical plan operators to generate camera directive sequence for specific type of action sequences. The camera directives are translated into constraints. Then a geometric constraint solver implemented in the underlying game engine will find a camera placement solution which satisfies the constraints.

The authors describe the formalization of film idioms. The approach was implemented using a service-oriented architecture called Mimesis for the control of narratives in a virtual world. The proposed camera planning algorithm uses a de-compositional partial-order casual link planning system called Longbow which is implemented in LISP. The camera control was tested within the Unreal Tournament 2003 game engine. The authors observe that the approach may not work at real time for large stories because it needs the story details for pre-planning camera movement. They claim that they have demonstrated how the effective elements of a story are communicated by generation of communicative plans by a discourse planning algorithm through a plan space search.

The authors identify the following future work:

- To generate automatically the annotations of the story plan with additional information about the action and characters from knowledge of plan structure.
- To optimize geometric constraints solver by utilizing high level context aware directives to reduce search space.
- To develop strategies for comparative empirical evaluation of cinematic camera planning systems.

17. Kumar, V. (1992) Algorithms for Constraint Satisfaction Problems: A Survey. *AI Magazine,* **13(1)** 32-44.

The problem addressed in (Kumar 1992) is how to make solution techniques of CSP more accessible to the readers. The paper has dealt with first assignment to the variables which satisfy all of the constraints. The scope of discussion of the paper is further limited to the solution techniques for a class of CSP called binary CSP where each constraint is either unary or binary. A unary constraint involves a single variable and a binary constraint involves two variables. The author provides a brief overview in tutorial format

of many solution techniques utilized to solve the problems in AI and other Computer Science areas cast as a special case of the CSP.

The paper is a survey of solution techniques of CSP. At first, the generate-and-test paradigm is discussed. Then more efficient backtracking algorithms are discussed. But the algorithms have also exponential complexity. Thrashing and redundant work performed are identified as two major drawbacks of the standard backtracking algorithm. Node inconsistency and lack of arc consistency are described as two main causes of thrashing. Node inconsistency arises whenever the domain of a variable contains a value that does not satisfy a unary constraint on the variable. The author suggests that node inconsistency can be eliminated by removing all the values from the domains of each variable that do not satisfy a unary predicate and arc consistency can be achieved by making each arc of constraint graph consistent. The following three techniques are identified to avoid redundant work and improve performance of the backtracking algorithm:

1. Constraint propagation at search node.
2. Reason maintenance or intelligent backtracking.
3. Ordering of variable and instantiation of different values.

Using the constraint propagation technique, a given CSP can be transformed into another CSP which has a smaller search space compare to the original one in such a way that the same failures are not encountered again. After describing AC1, AC3, and k-consistency, the author investigates the conditions for the existence of search order that is backtracking free. Then combination of constraint propagation and backtracking techniques is discussed. The author states that only a limited form of constraint propagation with backtracking can be applied for better results. Next, intelligent backtracking and truth maintenance techniques are discussed. Using these techniques, information about a failure is kept updated and used during search of the remaining search space. Truth maintenance methods deal with developing general techniques to attach justifications or assumptions to the inference. At last, approaches using variable and value instantiation ordering techniques are discussed. The author observes that application of variable ordering techniques will move failures to upper levels of the search tree. On the other hand, application of value instantiation ordering techniques will make a solution of CSP moved to the left of the search tree. The author claims that thrashing of the backtracking algorithm can be totally eliminated with much more cost than that incurred by the simple backtracking if any of constraint propagation, reason maintenance or intelligent backtracking, or variable ordering or value instantiation ordering techniques is applied to an extreme. On the other hand, a simplified version of the techniques can be used together with backtracking to reduce overall search space efficiently. But the optimal combination of these techniques will be problem dependent.

The author identifies future work as to find an optimal combination of various improvement techniques of backtracking algorithm for different problems.

18. Languénou, E., Benhamou, F., Goualard, F., and Christie, M. (1998) The Virtual Cameraman: an Interval Constraint Based Approach. In *Constraint Techniques for Artistic Applications (Post ECAI'98 Workshop),* Brighton, UK.

The problem addressed in (Languénou et al 1998) is how to help graphic artists create camera movements easily and relieve them from the need of understanding mathematical concepts underling the camera movements. The authors begin by referring to their first implementation of the work in (Jardillier and Languénou 1998) which used an interval solver based on recursive subdivision of the search space and state that the first implementation of the method was not efficient enough to tackle complex camera movements. Like a user of the first implementation, a user of the present implementation of the method will be able to define constraints on the image space, on the objects of the scene, or on the both. The authors used Declarative Language with Interval Constraints (Declic) which utilized two solvers for enforcing hull-consistency over some primitive constraints and box-consistency over global complex constraints. The more sophisticated constraint solving techniques are employed to handle non-linear constraints aroused from modeling of camera movements and screen-space constraints. Interval computation is used in the solver to avoid key-framing and interpolation. An extension of the core local-consistency algorithm is used to process universally quantified time constraints. The solver will process global camera movements for the complete animation. It will generate a set of camera movement satisfying user defined constraints from which the users will be able to select camera movements to create desired animation.

The authors describe mathematical models of the scene objects, camera, and their movements. Constraints used in the paper are similar to the constraints used in (Jardillier and Languénou 1998). They also describe some of the constraints on the camera, the projected objects, and objects of the scene in Declic. The camera control method was under implementation. A prototype was implemented in Declic. The authors state that the users can devise new image space constraints easily using the approach and satisfaction of the specified constraints can be characterized precisely. They claim that the final implementation will minimize amount of repetitive tasks for the users and will require less image representation model knowledge of the users.

The authors identify the following future work:

- To improve the resolution process to increase number of degree of freedom of the camera.
- To use a high level constraint-oriented language for flexibility of development and easy maintenance.
- To devise a proper way to deal with constraints that must hold in a given interval as a built-in process in Declic.
- To demonstrate usability of the tool for the users having no knowledge of camera movement mathematics.

19. Lin, T., Shih, Z., and Tsai, Y. (2004) Cinematic Camera Control in 3D Computer Games. *The Twelfth International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2004, WSCG 2004,* University of West Bohemia, Campus Bory, Plzen-Bory, Czech Republic, 289-296.

The problem addressed in (Lin, Shih, and Tsai 2004) is how to apply basic cinematic camera control in 3D computer games to enhance the viewer's gaming experience. The authors begin by referring to (Drucker 1994), (Drucker and Zelter 1995), (He, Cohen and Salesin 1996), and (Halper, Helbing, and Strothotte 2001) and state that the camera control concepts of the above mentioned papers are used in this paper. They also refer to (Drucker 1994), (Drucker and Zelter 1995), (Bares, Thainmit, and McDermott 2000), and (Halper, Helbing, and Strothotte 2001) and state that the solver concepts of the above-mentioned papers are used in this paper. The proposed event-driven computer game system with cinematic camera control consists of a real-time application, a camera control system, and a renderer. The camera control system consists of camera modules and descriptions of shots. A camera module represents a cinematographic shot. It solves the constraints defined by the shot. It handles camera placement and transition based on user input, its constraint list, and parameters from the application. The system has eight camera modules to handle any situation. A description of a shot is a sequence of shots. It represents a cinematographic scene and is a finite state machine of camera modules. The descriptions of shots are organized hierarchically. At each time tick, the game application will send the camera control system a list of parameters which are relevant for the time tick. The camera parameters will be handled by the active camera module. By using the parameters and the current state of camera setting, the camera control system produces camera specification for the time tick and sends the specification to the renderer. At the same time tick, the game application supplies the renderer with content of the game which includes static geometry, materials, light, and the story. The renderer uses the information for the same time tick from both the game application and the camera control system to display.

The authors describe some cinematographic principles and explain various descriptions of shots using their camera modules. Frame coherence and obstruction avoidance systems are described. If frame coherence is required, camera position and angle are interpolated. For obstruction avoidance, a line from camera to the bounding sphere of the actor is used to determine if there is an obstruction or not. In addition, if smooth camera movement is required, the line intersection test is replaced with a cylinder to detect an obstruction before it happens. Constraints and constraint solver used in the paper are same as those used in (Halper, Helbing, and Strothotte 2001). The system was implemented in C++ and based on Half-life game engine. The test environment was a Pentium 4, 2133MHz processor with 256MB RAM PC. The authors explained improvement of few captured shots using the camera modules. The authors claim that the camera control system can generate shots automatically and arrange the shots to create cinematic effect which is suitable for 3D computer game.

The authors identify the following future work:

- To generalize the camera system to fit in to any graphic engine in place of current Half-Life game engine.
- To integrate more techniques like emotional state to extend the ability of the system.
- To consider the use of expert system to deal with complex camera planning due to too many conditions.

20. Nadel, B. A. (1990) Constraint satisfaction algorithms. *Computational Intelligence,* **5(4)** 188-224.

The problem addressed in (Nadel 1990) is how to make the complexity of the solution algorithms of CSP more understandable to the readers. The author starts by referring to the preliminary version of the paper published in 1988 as 'Tree search and arc consistency in constraint satisfaction algorithms' and states that several more algorithms have been added in the current version of the paper and the presentation of the paper has also been significantly restructured. The paper has dealt with all the solution assignments. The author has conducted what he called a unified survey on the solution algorithms based on tree search, arc consistency, and hybrids of the two methods. An arc consistency algorithm makes a directed arc (x, y) of a constraint graph consistent by removing all the values from the domain of the variable x for which there is no corresponding value in the domain of the variable y to satisfy the constraint C(x, y). Both parameterized full and partial arc consistency algorithms are introduced. Parameterized arc consistency procedures can be combined with tree search algorithms. Hence arc consistency processing can be done on the sub problems rooted at each individual search tree node. The author observes that implication of non parameterized arc consistency algorithms are either the algorithms are adequate to solve a CSP instance or can only be used for preprocessing before application of a search algorithm to solve a CSP instance.

The paper is a survey of solution algorithms of CSP. Tree search solution algorithms Backtracking, Backjumping, and Backmarking are discussed. The author observes that several important algorithms such as classic backtracking, and Haralick's full and partial Lookahead algorithms when rearranged take hybrid form. But all the arc consistencies of rearranged algorithms do not achieve full arc consistency at the tree nodes. In the spirit of full arc consistency procedures such as AC1, AC2, and AC3, new partial AC procedures have been called AC⅕, AC¼, AC⅓, and AC½ based on the degree of partial arc consistency achieved by the algorithms. All of the arc consistency algorithms known at that time are discussed. Then nine hybrid algorithms with full arc consistency TSRACi, TSACi, and RFLi (i = 1, 2, 3) and four hybrid algorithms with partial arc consistency FL, PL, FC, and BT are discussed. The n-queens and confused n-queens problems were formulated as complete, binary CSP instances to compare the complexity of fifteen tree search and hybrid algorithms empirically for all the solution assignments. A tree searching algorithm called Backmarking by Gasching and a hybrid algorithm called Forward Checking by Haralick are reported as most efficient ones. Nadel observes that the arc consistency processing at the nodes for hybrid algorithms increases number of nodes. Hence the methods using arc consistency should sufficiently reduce average work

per node so that less work over the whole tree is needed. The author claims that little arc consistency by hybrid algorithms reduce workload over whole tree and unified view taken for the survey suggests several new algorithms. One of the new algorithms is considered as the best one based on preliminary test results. The new algorithm has been named tentatively as ‘TSSTAC3’ for ‘Tree Search + Singleton Target Node AC3’.

The author identifies the following future work:

- To find new applications of CSP.
- To understand theoretically and empirically existing solution algorithms of CSP.
- To develop new solution algorithms of CSP.
- To generalize the problem and its algorithms.

Appendix B: STATEMENT OF ATTESTATION

The survey is completely my own work.

Mohammed Liakat Ali