2008

Control of constraint weights for an autonomous camera

Md. Shafiul Alam
University of Windsor

Control of Constraint Weights for an Autonomous Camera

by

Md. Shafiul Alam

A Thesis
Submitted to the Faculty of Graduate Studies
through School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada

2008

Your file Votre référence
ISBN: 978-0-494-47013-8
Our file Notre référence
ISBN: 978-0-494-47013-8

Author's Declaration of Originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

Constraint satisfaction based techniques for camera control has the flexibility to add new constraints easily to increase the quality of a shot. We address the problem of deducing and adjusting constraint weights at run time to guide the movement of the camera in an informed and controlled way in response to the requirement of the shot. This enables the control of weights at the frame level. We analyze the mathematical representation of the cost structure of the domain of constraint search so that the constraint solver can search the domain efficiently. We start with a simple tracking shot of a single target. The cost structure of the domain of search suggests the use of a binary search which searches along a curve for 2D and on a surface for 3D by utilizing the information about the cost structure. The problems of occlusion and collision avoidance have also been addressed.

Dedicated to my father late Md. Khorshed Alam

Acknowledgements

I would like to thank my supervisor Dr. Scott D. Goodwin, Professor, School of Computer Science, University of Windsor for his guidance, valuable advice and patience that helped me successfully conduct my research and prepare this thesis. I am particularly grateful for the critical analysis, valuable suggestions and editing made by my external reader Dr. Ron M. Barron, Professor, Department of Mathematics and Statistics, University of Windsor that have greatly enhanced this thesis, and for his continuous support throughout my graduate studies at the University of Windsor.

I would also like to thank my internal reader Dr. Asish Mukhopadhyay, Professor, School of Computer Science, University of Windsor and my thesis committee chairman Dr. Richard Frost, Professor, School of Computer Science, University of Windsor for their contributions to my thesis.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

Constraint satisfaction based technique to camera control is a general approach. Complex properties can be easily represented as constraints. There is no limit to the number of properties that these techniques can handle. It has the flexibility to analyze the space of possible camera parameters and find a suitable solution in interactive dynamic environments (Christie and Olivier, 2006; Bourne and Sattar, 2004b; Bares, Thainimit and McDermott, 2000).

However, because of conflicting requirements in the problem domain the resulting constraint satisfaction problem is an over-constrained problem (Bourne and Sattar, 2004b). Constraint weighting is used to give each constraint a priority order and the weighted sum of the costs of violation for all the constraints is used as the objective function. Constraint satisfaction optimization is used to solve the problem.

Bares, McDermott et al. (2000), Bares, Thainimit and McDermott (2000) and Bourne and Sattar (2005a) use user specified constraint weights based on the relative importance of the constraints in a particular type of shot. Bourne and Sattar (2005b), and

Bourne (2006) use example animation trace to deduce the weights for all the constraints corresponding to that animation. The resulting constraint weights and the desirable values for all the constraints are coupled together, and they are appropriate for a particular type of shot that is equivalent to the example animation trace. In our approach the physical significance of the weights are used to deduce them automatically from the requirements of the shot.

It seems that Bourne and Sattar (2006) and Bourne (2006) were the first to identify the higher cost regions of the search domains for different weights. But they use only height, distance and orientation constraints, and their work is limited to either equal weights or only one constraint having higher weight. To take advantage of this information about the search space they use a specialized constraint solver called a "sliding octree solver" to search the domain quickly. It prunes the regions with poor solutions quickly. But, until now, the mathematical representation of the cost structure of the domain of search has not been studied. Consequently, the information about the structure of the domain could not be utilized to arrive at an exact solution or to direct the search by that information. We use the cost structure of the domain to search it. This will ensure an effective and informed way of pruning the domain, and there will be no problem with local minima.

According to Halper et al. (2001), one of the main challenges of the camera control problem is to find a balance between optimal camera positions and frame coherence. They do not use constraint to enforce frame coherence. On the basis of the prediction about the future target positions, they evaluate future camera positions and move the camera toward those positions. Bourne and Sattar (2004b) say that this does not

ensure frame coherence and the method is fully dependent on the algorithm used to predict the movement of the target. Bourne and Sattar (2005a) and Bourne (2006) introduced the frame coherence constraint in the weighted constraint representation of the camera control problem. Since the properties of the frame coherence constraints are different from those of visual constraints, we decouple the two types of constraints and use their relative priority to influence the search of the solution camera position.

1.2 Structure of the Thesis

Chapter 2 discusses the current research in autonomous camera control using the constraint satisfaction technique. Chapter 3 presents our analysis of the weighted constraint representation of the camera control problem and its solution technique. Chapter 4 extends the method to handle the problem of occlusion and collision avoidance. Chapter 5 contains the conclusions and direction for further extension.

Chapter 2

Automatic Camera Control

2.1 Applications of Camera Control in Virtual Environment

The theory of camera control is used in nearly all 3D interactive applications. Recent development in 3D graphics technologies has created a great opportunity for a new generation of interactive 3D entertainment, education, and simulation based training systems (Bares and Lester, 1999a, 1999b). Applications of automatic camera control are:

- Computer and video games
- Generating 3D illustrations
- Producing 3D animated movies
- Generating 3D animated explanations to achieve communicative goals
- Visualizing museum walkthroughs and virtual chatrooms
- Generating 3D scenes with simulated humans for ergonomic simulation and virtual reality training

2.2 Camera Control in Computer Games

Barwood (2000) says, "Games may owe something to movies, but they are as different from them as movies are different from theater." Christie et al. (2005) say that since most games are controlled by a player and define a dynamic environment, camera control in computer games is more complex than that for a static environment or for the areas where the evolution of action is known in advance.

Halper et al. (2001) explain that camera techniques in computer games will have to use their own languages and rules on top of cinematographic techniques. They argue that unlike film or theatre, computer games are highly interactive. This makes the rehearsal or staging of actions in computer games impossible before actual shooting. So, they infer that cinematographic techniques can not be used in computer games since they depend to a large extend on trial and error. They also argue that for real movies it is possible to change the position and orientation of actors, the scene and even the script, but that is not possible in computer games. In the latter case the scene has to be shot as it has evolved up to that point of time even if it results into a poor quality shot. Thus, they deduce that converting cinematographic techniques into idioms for positioning and moving the camera in computer games will not be adequate to convey the visual information in the interactive situation that characterizes the computer games.

Christie et al. (2005) say that real-time camera control in computer games in particular, and computer graphics applications in general, is similar to that of documentary movies in that the position and orientation of scene elements must not be changed by the camera module. On the other hand Hawkins (2005) says that in some

cases cheating is necessary, and he suggests doing it, e.g., adjusting the position of a candle that is placed to enhance the mood of the scene only.

Christie et al. (2005) say that although the camera system is increasingly becoming the decisive factor for the success of computer games, little attention has been given to it. They also note that although editing can enhance storytelling and reduce confusions that are seen in many games, they are used very rarely. They note that Tomlinson et al. (2000) and Friedman and Feldman (2004) have used some common editing techniques to effectively engage the player, but their editing is based on the nature of action in the environment rather than on the emotion of the player.

Christie and Olivier (2006) argue that the camera system is only a small part of a computer game engine, and only a small part of the time between successive frames can be devoted to it.

Christie et al. (2005) and Christie and Olivier (2006) classify camera systems in computer games into three main categories:

- First Person Camera Systems: Users control this type of camera. They see through the character's eyes. It gives the user the feeling that he is the character or the game and that he moving around in the virtual environment. Doom and Quake are among the many games that use this type of camera.
- Third Person Camera Systems: This type of camera system tracks the target from some fixed positions and changes the camera's position and orientation in

accordance with the environment (such as occlusion problem) and the target's interaction with the environment.

- Action Replay Camera Systems: Used to highlight important events such as crashes in driving games, goals in football games, etc.

2.3 Properties of Camera

A camera has 7 degrees of freedom, viz., 3 for position, 3 for orientation and 1 for field of view. Properties of a shot determine these parameters of the camera. There are three types of properties, such as on-camera, on-screen and on-path properties (Christie and Olivier, 2006). These are discussed in the following subsections.

2.3.1 On-Camera Properties

These properties constrain the camera parameters more or less directly. Vantage angle, focal length, distance and collision avoidance are examples of these properties. Distance and vantage angle are the most important properties (Katz, 1991) and as such are considered by most researchers, such as Drucker (1994), Bares, Thainimit and McDermott (2000), Halper et al. (2001) and Christie and Languenou (2003). Collision avoidance is addressed by a few researchers such as Bourne and Sattar (2005a) and Christie and Languenou (2003). Focal length is directly specified by some approaches such as Blinn (1988) and He et al. (1996). Focal length is determined by constraints in some other approaches such as Drucker (1994) and Christie and Languenou (2003).

2.3.2 On-Screen Properties

These properties specify the location of the objects on the screen in a relative, absolute or approximate manner; and other properties such as size, orientation and occlusion of objects. Location of objects on screen are specified absolutely by Blinn (1988), Gleicher and Witkin (1992), Drucker (1994), He et al. (1996) and Halper et al. (2001). Relative position of objects on screen are specified by Olivier et al. (1999), Bares, McDermott et al. (2000), Christie and Normand (2005) among others. Approximate locations of objects on or out of screen are specified in such papers as Jardillier and Languenou (1998), Olivier et al. (1999), Bares, McDermott et al. (2000), Halper and Olivier (2000), Pickering (2002) and Christie and Languenou (2003). A very large set of on screen properties are considered by researchers.

Occlusion is a geometrical relation (Christie and Olivier, 2006). An object is occluded if some part of its image is obscured by the image of another object. An object may be totally occluded, partially occluded or unoccluded. Some occlusion avoidance techniques are described below:

- Ray casting: In this technique a ray is cast from the camera to the target. This technique is simple and efficient; and is used in most computer games (Christie and Olivier, 2006). To improve performance, the ray intersection is found with the bounding volume instead of the target.
- Consistent regions: Bounding boxes of the potentially occluding objects are projected onto the sphere with centre at the centre of each of the subjects for which the occlusion avoidance is being considered. All these projections are

converted into a global spherical coordinate system and negated to obtain the occlusion free region (Bares and Lester, 1999a, 1999b).

- Hardware rendering: In this technique the scene is rendered in the hardware stencil buffers. Each object is shown with separate colour. The buffer is read back and analyzed to generate the occlusion information (Halper and Olivier, 2000; Halper et al., 2001).
- Bounding volume: In this technique bounding volumes are computed around both the camera and the target. Objects are not allowed to enter these bounding volumes (Marchand and Courty, 2002).

2.3.3 On-Path Properties

Properties in this category include frame coherency, collision avoidance, etc. Frame coherence requirement was first proposed by Halper et al. (2001), but they did not consider frame coherence as a constraint. They use prediction of target movement and guess where the camera should be in future frames and move the camera towards that position. Bourne and Sattar (2004b) argue that this method does not guarantee frame coherent movement of the camera, because it entirely depends on the algorithm for predicting the future movement of the target. Bourne and Sattar (2004b) use two constraints such as frame coherence distance and frame coherence rotation to represent the requirement of frame coherence. In their approach these two constraints must be satisfied to some degree, otherwise the solution is identified as invalid.

Bourne and Sattar (2004b) indicate that the method proposed by Christie et al. (2002) represents the camera movement as a constrained combination of predefined

cinematographic primitive motions such as panning, tracking, dolly, traveling, arcing and zooming. It ensures frame coherence to some extent and the camera's motion can be reasonably accurately represented. But the predefined motions do not have the dynamic property that the constraints can provide, and they do not support unexpected movements required by the camera, if those movements are not represented as predefined motions. Bourne and Sattar (2004b) say that the constraints completely determine the motion of the camera in their approach.

2.4 Requirements of Camera Control

Some of the design guidelines for a general purpose autonomous camera control system as identified by different researchers are given below (Bares and Lester, 1999a, 1999b; Bares, McDermott et al., 2000; Bares, Thainimit and McDermott, 2000; Bourne and Sattar, 2005a):

- User-Specified Viewing Goals: The camera should be able to deal with the user specified viewing goals (Bares and Lester, 1999a, 1999b).
- Environmental Complexity: The camera must be able to operate in an arbitrary environment (Bares and Lester, 1999a, 1999b).
- World Non-Interference: The camera system should not modify the world to simplify the problem (Bares and Lester, 1999a, 1999b; Bares, Thainimit and McDermott, 2000). In some cases cheating may be acceptable, e.g., adjusting the position of a candle that is used to enhance the mood (Hawkins, 2005), cutaway of occluders (Feiner and Seligmann, 1992).

- Failure Handling: The camera should be able to produce the next best solution when a satisfactory solution is not available, so that the users get an acceptable view of the scene for those situations (Bares, Thainimit and McDermott, 2000).
- Autonomy: The camera must be able to move autonomously without user intervention. While moving, the camera must be able to maintain the visual properties (Bourne and Sattar, 2005a).
- Reactive: The camera module must be able to work without any prediction about the future position of the target (Bourne and Sattar, 2005a).
- Real-time: The camera system must be able to perform all its computations in real-time so that it can be used in interactive applications (Bares, McDermott et al., 2000; Bourne and Sattar, 2005a).
- Dynamic: The camera system must be able to operate in a dynamic environment and with dynamically changing targets (Bourne and Sattar, 2005a).

2.5 Approaches to Camera Control

There are many approaches to solve the problem of autonomous camera control. These approaches have o riginated from many diverse disciplines such as medical imaging, robotics and virtual cinematography. They are given below:

- Predefined camera position relative to the subject
- Idiom-based System
- Constraint Satisfaction Approach
- Automated Camera Control Assistant
- Automated Camera Navigation Assistant

- Potential Fields
- Intelligent Agent
- Image-based Visual Servoing
- Spline Systems
- Path Planning

2.6 Camera Control Using Constraint Satisfaction Techniques

In the constraint satisfaction approach the user describes the scene in terms of cinematographic properties (Mascelli, 1998; Arijon, 1991; Katz, 1991). The properties are expressed as numerical constraints and/or objective functions on the parameters of the camera. The constraints must be satisfied and the objective functions are to be maximized or minimized. The problem is solved by exploring the space of camera parameters to minimize / maximize the objective function while satisfying the constraints (Christie et al. 2005; and Christie and Olivier, 2006). The search may be broadly categorized into two groups: complete and incomplete.

The camera has 7 degrees of freedom with each having a continuous large domain. Depending on the problem and its representation, the number of variables varies between 1 and 7 (Bourne and Sattar, 2004b). So, the search tree is very wide and not very deep. Due to the shallow depth of the search tree for the camera control problem, the search heuristics for complete search, such as back jumping or conflict directed back jumping, provide little benefit in pruning the large space. Bourne and Sattar (2004b) say that although local search can find a reasonable solution to a large and difficult problem easily, it does not guarantee the best solution and it has the tendency to violate the frame

coherence properties more often than the complete search because of its use of the property of randomness in its solution method. To increase the probability of finding the best or near-best solution, it is necessary to evaluate a maximum number of solutions. But this reduces the performance. So, the authors suggest using a trade-off between performance and accuracy.

The constraint satisfaction based approaches differ in the nature of domains considered for search. They vary from the consideration of fully continuous to fully discrete domains. The discrete domains are derived by regular or stochastic subdivision of the domain of camera parameters. The approaches also differ in the consideration of the solving techniques. They vary from pure optimization based techniques to pure constraint satisfaction problem based techniques. Pure optimization based techniques use soft constraints and try to find the best solution with respect to an objective function. Pure constraint satisfaction problem based techniques use the hard constraint approach and perform exhaustive search on the domain.

2.6.1 Domain Selection Strategy

Bares, McDermott et al. (2000) limit the domain of each parameter of the camera between the allowable minimum and maximum values of the constraints. For each constraint they determine the valid region of space for the related camera parameters. The common valid region is obtained by taking the intersection of all the valid regions. The solver searches in the common valid region at discrete steps on all the 7 camera parameters. The search starts at a relatively coarse grid, then recursively searches at increasingly finer resolution.

But the resulting search space is still coarse and the method does not use the speed of the camera to scale it (Bourne and Sattar, 2004b).

Bourne and Sattar (2004b) use a dynamic domain selection strategy that is based on the movement of the camera in the past frame. The resulting domain will have fine resolution for a slowly moving camera and coarse resolution for a fast moving camera.

2.6.2 Using Information about the Search Space

In Bourne (2006) and Bourne and Sattar (2006) the authors analyze the cost structure of the domain of search of the constraints such as distance, height and orientation (for a tracking shot) for different weights. They show that the optimal solution is often in the middle of the domain. But the movement of the target and the use of different constraints move the optimal solution to various regions of the search space. From this information they infer that there is no guarantee that the solution will be around the mid-point or that starting from the mid-point will find the optimal solution quickly. Using the information about the search space for the weighted constraint representation of the camera control problem, the authors have identified some design goals for the constraint solver that are given below:

- Search large domains quickly as there are large regions of space with poor solutions.
- Exploit the spatial characteristic of the application.
- Utilize the gradient of the domain to focus near the optimal solution.

- Must be able to find the optimal solution at arbitrary place because there is no prior knowledge about the location of the optimal solution.

They propose a sliding octree solver and claim that it fulfils the above design goals. They claim that 120 searches are sufficient to find an optimal solution.

2.7 An Overview of Constraint Satisfaction Based Approaches

The following approaches are used for the incomplete search:

- Constraint Satisfaction Optimization Problem Based Approaches: Drucker (1994), Drucker and Zeltzer (1994, 1995), Bares, McDermott et al. (2000), Christie and Normand (2005), Bourne and Sattar (2005a, 2005b) and Bourne (2006) use the constrained optimization based technique.
- Pure Optimization Based Approaches: Olivier et al. (1999), Halper and Olivier (2000) and Pickering (2002).
- Partial Constraint Satisfaction Problem Based Approaches: Halper et al. (2001) and Bourne and Sattar (2004a, 2004b) use a hierarchical constraint approach.

The complete search approaches use the following techniques:

- Partial Constraint Satisfaction Problem Based Approaches: In this approach some of the constraints are considered as hard and others as soft. Bares et al. (1998), Bares and Lester (1999a, 1999b) and Bares, Thainimit and McDermott (2000) use this approach with hierarchical constraints.

- Pure Constraint Satisfaction Based Approaches: In this approach all the constraints are considered as hard. Jardillier and Languenou (1998), Languenou et al. (1998), Christie et al. (2002) and Christie and Languenou (2003) use this approach with pure interval arithmetic.

Drucker (1994) and Drucker and Zeltzer (1994, 1995) offer the first constraint based camera control system. It is called CAMDROID. It is a task based camera model which specifies the behaviour of the camera using task level goals and constraints on the camera parameters. They regroup some of the cinematographic primitives into camera modules which are similar to shots in cinematography. The camera module provides an interface for user interaction with the module. The constraints of a module are combined by a constraint solver. They propose a constrained optimization solver based on Feasible Sequential Quadratic Programming to find a solution.

This paper is the first to handle the screen-space constraint in cinematographic context, and it presents a complete set of screen-space constraints (Jardillier and Languenou, 1998). Jardillier and Languenou (1998) also note that the method computes a static camera solution for each frame, and hence interpolation between key frames is of no use because of the risk of constraint violation. Christie and Olivier (2006) say that the method is limited to a smooth fitness function and smooth constraints which is difficult to ensure in computer graphics. They also note that the method is prone to local minima and is sensitive to initial condition. Bares et al. (1998) argue that it does not offer a systematic solution for constraint failure that can frequently happen in a dynamic virtual environment with complex scene.

In Bares et al. (1998) and Bares and Lester (1999a, 1999b) the problem of constraint failure is addressed by using the partial constraint satisfaction problem technique of relaxing weak constraints in the order of lowest priority to provide the user with an approximate solution when constraints fail and, if necessary, decompose a single shot into multiple shots. They have implemented their method in CONSTRAINTCAM, a framework for camera control.

Their camera system implements four types of constraints such as vantage angle, shot distance, occlusion avoidance and subject inclusion. Each of these constraints can be applied to any object. Each of them has a relative strength and a marker which shows if the respective constraint can be relaxed.

The constraint solver first identifies the consistent regions of subject space relative to each constraint of every subject. The consistent region for each constraint of every subject is expressed in terms of its local spherical polar coordinate system with the origin at the mid-point of the respective subject. Then, each local consistent region is converted into a common global spherical polar coordinate system. The origin of this global coordinate system is placed at the centre of all subjects of interest. The constraint solver then attempts to find the intersection of all the consistent regions. If the intersection of the consistent regions is non-empty then the constraint solver searches for the optimal vantage (θ, φ) within the intersection that is nearest the optimal vantage angle of all the subjects. Then the distance of the camera from the centre of the subjects is determined by taking intersection of the distance intervals of the consistent regions related to the viewing distance constraints of the subjects. If the optimal vantage angle is occluded then the distance is decreased to place the camera in front of the nearest occluder. For

occlusion detection, a ray casting on bounding volumes of the occluders is used. This determines the solution camera position.

If no solution is found, the solver first tries to find an approximate solution which satisfies as many higher priority constraints as possible by satisfying the constraints in the order of decreasing priority. To do so, it first identifies the incompatible constraints by constructing an "incompatible constraints pair graph". Each node in the graph represents a constraint. Pairs of inconsistent constraints are joined by an arc. Then it repeatedly relaxes the weak constraints in the order of increasing priority until all the inconsistencies are resolved or no more relaxation is possible.

If the relaxation is successful, a single shot solution is determined. Otherwise, it decomposes the original problem into a minimum number of sub-problems by employing the strategy of satisfying as many constraints as possible in each sub-problem. It then displays the multiple shots either sequentially or simultaneously using a composite shot consisting of a main viewport and one or more insert viewports.

Christie and Olivier (2006) claim that CONSTRAINTCAM is based on only a limited subset of cinematographic properties and is applicable to relatively small problems involving only two subjects. They also note that a drawback of using the partial constraint satisfaction technique is that the user has to provide a hierarchy of constraints, but determining the hierarchy on the basis of visual appearance is not always trivial. Bourne (2006) argues that the process of relaxing the constraints repeatedly can be time consuming if the number of subjects and constraints increases. Halper et al. (2001) say

that CONSTRAINTCAM uses purely reactive application of constraints which results in jumpiness of the camera.

Bares and Lester (1999a, 1999b) have extended the approach of Bares et al. (1998). They assigned relative priority to the targets also. The strength of a constraint is calculated by multiplying the priority of the constraint and that of the subject involved. To remove inconsistency in the incompatible constraints pair graph, this measure of strength is used. When producing a multi-shot solution, if any inset shot is not better than the overview shot (according to the total cost of the strengths of the failed constraints) that inset shot is removed. The authors claim that CONSTRAINTCAM performs in real-time. They also noted that it misses some solution positions which are in front of the occluding objects.

Bares, Thainimit and McDermott (2000) is an extension to the previous work on the CONSTRAINTCAM. It supports fifteen different types of constraints that can be applied either on the camera parameters or on one or two objects in the image. Each of the constraints has a range of allowable values, an optional optimal value and a relative priority value with respect to other constraints. A weighted sum is used to measure the constraint satisfaction at each potential position in the domain of the camera parameters. The cost of each constraint is evaluated on the basis of its nearness to its respective optimal value and its range is specified as 0.0 to 1.0. The weight is the relative priority of the respective constraint. The weighted costs of all the constraints are added to obtain the cost for a potential solution.

An exhaustive generate-and-test strategy is used to search the domain of all the camera parameters at discrete steps and test its satisfaction using the weighted sum measure. The one with the highest value is returned as the solution for the camera position, orientation and field of view. It searches at steps of 20x20x20 grid on camera positions, 15 degree intervals for orientation parameters and 10 units for field of view. It has been noted that the exhaustive generate-and-test method is impractical.

Bares, McDermott et al. (2000) further extends CONSTRAINTCAM. In this extension they propose a heuristic search algorithm which is also applied at discrete steps like the previous one. It uses the same 15 different types of constraints and the same weighted sum measure for evaluating the fitness of the potential solution.

The heuristic search method uses the allowable minimum and maximum values of constraints to reduce the size of the search domain of the 7 parameters of the camera. For each constraint it determines the valid regions of space of the respective camera parameters. Common valid region for all the constraints are evaluated by intersection. The common valid region is searched at discrete steps on all the 7 parameters of the camera using a generate-and-test method. The search begins by using a relatively coarse grid, then recursively searches at increasingly finer resolution. At each step a pre-specified number of best candidate solutions are selected and the search continues recursively about those candidate solutions at finer grid resolution. The process stops when a solution is found that exceeds the given minimum threshold value for the weighed total fitness value or when the minimum grid resolution is reached.

In the first iteration a 9x9x9 grid is scanned. The second iteration is scanned over a 5x5x5 grid about 5 best potential solutions from the first iteration. Occlusion is tested using the ray casting method. They claim that the solver computes a shot in 0.016 milliseconds, whereas an exhaustive search algorithm takes 20 to 30 minutes to compute 2 million shots.

Christie and Normand (2005) extend the idea of identification of the feasible region of space before search (Bares, McDermott et al., 2000; Pickering, 2002). The method provides semantic meaning to each valid volume with respect to its corresponding constraint in terms of cinematographic properties. The volumes are intersected to obtain the feasible region of space for search. If the intersection is empty the problem has no solution. If there is a solution, stochastic local search is used to find a minimum cost solution with respect to a cost function. The solver starts with an initial guess about the solution from inside the feasible volume. In each iteration it searches a set of neighbours around the current configuration of the camera and keeps the best one as the centre of the next iteration. The search ends when a solution is found or after a predefined number of steps. Finally, the user can interact with the system and utilize the semantic information of the volumes.

This approach can be applied to a static camera only because of the computational cost involved in finding the intersection of the volumes and the dependency of the computation of the volumes on the object's position in space (Christie and Olivier, 2006).

Halper and Olivier (2000) present a camera control framework called CAMPLAN. They say that the approaches offered by Blinn (1988), Seligmann (1993),

Drucker (1994), Christianson et al. (1996), Bares et al. (1998) and Bares and Lester (1999a) have the limitations of limited range of image properties that may be specified and that they use unrealistic point-based characterization of scene elements. These shortcomings are addressed in their proposed approach. They use genetic algorithms to find the optimal camera position where all the degrees of freedom of the camera are encoded in the chromosomes of each gene. They propose an "optimized visible surface determination algorithm" to evaluate occlusion constraints. Christie and Olivier (2006) argue that the method is computationally costly and has non-deterministic behaviour.

Pickering (2002) uses constraints on the camera position to find a valid region. In the second step, a genetic algorithm is used to search in that region. Christie et al. (2005) say that this method prunes the search space efficiently and results in searching in the interesting regions. Christie and Olivier (2006) point out that the main problem is that it is difficult to model multiple constraints into a single objective function. They also note that the values of the weights are usually determined by a tedious generate-and-test method.

In Halper et al. (2001) the authors extend the approach of CAMPLAN (Halper and Olivier, 2000). They introduce the idea of frame coherence (Bourne, 2006). Halper et al. (2001) refer to Drucker et al. (1992), Drucker and Zeltzer (1994, 1995) and Bares and Lester (1999a) and note that none of them use predictive analysis of the virtual environment, or impose constraints based on present camera position and movement. They say that for these reasons they can not achieve a high level of frame coherence. They present what they claim to be the first constraint solver based on existing camera state and motion characteristics. Their constraint solver applies constraint hierarchically – first solves for certain constraints, then modifies the camera state to accommodate other

constraints. Each successive constraint minimally influences output camera state of its previous constraints.

Bourne (2006) says that the camera system of Halper et al. (2001) is only as good as its prediction system, that the computation of prediction in a complex environment can be expensive, and that allocating computation for predicting a future state that can never be reached is a waste of resources.

Bourne and Sattar (2004a, 2004b) use the following frame coherence and visibility constraints to ensure frame coherence and avoid occlusion, camera holes, camera cutting and unnecessary movement of the camera:

- Frame Coherence Constraints: Distance and rotation constraints are used to ensure frame coherence. Distance constraint ensures smooth displacement of the camera between frames by restricting the displacement of the camera to a distance that is based on the distance the camera has moved in the previous frames with some adjustments for acceleration or deceleration. Rotation constraint similarly ensures smooth rotation of the camera based on its rotation in the previous frames. Camera cutting is obviously avoided by the frame coherence constraints. Unnecessary movement is also avoided by the frame coherence constraints which cause the camera to remain at rest until the cost of the solution increases high enough to force the camera to move to another position to reduce the cost.

- Visibility Constraint: To avoid occlusion, non-visible areas behind occluders are given higher cost. To avoid camera holes, areas near the object increase the cost of the solution.

The authors use hierarchy of constraints in three layers where violation of higher layer constraints has higher costs. Level 1 is the highest layer in the constraint hierarchy. They consider frame coherence as the most important visual property, and so related constraints such as frame coherence distance and frame coherence rotation constraints are placed in this layer. They argue that the reason behind this choice is that unsmooth camera movement can cause motion sickness to the player. Level 2 has the constraints whose satisfaction is important but not mandatory. The visibility constraint which is related to the occlusion, camera cutting and camera hole is placed in this level since the authors consider smooth movement more important than these constraints. Level 3 contains the constraints which can be violated more readily in order to satisfy the constraints in the higher layers. The authors place the constraints related to the general behaviour of the camera in this level. They use a trailing camera to test their approach. They use distance and height constraints to implement the trailing camera and so place them in this layer.

The authors assign each constraint a cost that is proportional to its violation. The costs for all the constraints are added to obtain the total cost for a solution. They optimize the domain by specifying the upper and lower bounds to realistic values on the basis of the past movement of the camera. They use a branch and bound backtracking algorithm, and a greedy local search algorithm to search the domain. In the branch and bound algorithm the search continues into inconsistent regions of the search tree if the cost of

the partial solution is less than a specified maximum cost or the best cost solution found so far.

The local search algorithm initially assigns each variable a random value from its domain. It keeps this set of values as the best solution found so far. If the total cost of the potential solution is not zero, it randomly selects a variable and assigns a new value to it from its domain. It recalculates the total cost. If the cost is less than the previous best solution, the new solution replaces the previous best solution. The process continues iteratively until either a zero cost solution is found or a specified number of searches have been performed.

The performance of branch and bound is nowhere close to real-time, whereas the greedy local search performs close to real-time without using any optimization or search heuristics. The qualities of the solutions generated by the two algorithms are similar. Using these results they conclude that complete search is not necessary for this case.

Bourne and Sattar (2005a) is an extension of their earlier work (Bourne and Sattar, 2004b). They include a visibility constraint that influences the efficiency and structure of the constraint solver, and investigate application of the constraint weighting technique to the autonomous camera control problem. They describe the use of constraint costs and their weights to control the behaviour of the camera, and propose an efficient search heuristic for the solver.

The authors claim that prior to their paper the integration of the approaches to autonomous camera control and the methods of occlusion avoidance were not effective or

unified. Their goal is to achieve an effective and unified method by integrating constraint weighted local search for the solver and ray-casting for visibility.

These authors say that there are four major requirements for a successful autonomous camera, viz., autonomy, reactive, real-time and dynamic. They do not use any prediction for the future target states. This satisfies the requirement of a reactive camera. They claim that the minimal set of constraints required to adequately represent the visual properties of the camera and attain real-time performance are height, distance, orientation and frame coherence. This ensures automatic maintenance of visual properties of the camera (autonomy) and real-time performance. They use weighted average for the calculation of viewpoint for multiple targets. This addresses the requirement of a dynamic camera. The reasons behind the weighted average strategy for viewpoint are that the single target situation occurs most of the time and that the computational complexity involved in solving a multiple target situation as a constraint satisfaction problem is not justifiable in comparison to the frequency of its occurrence. Each of the height, distance and orientation constraints has a value and a user specified weight. The value of each of the weights is normalized with respect to the scale of values of the respective constraint.

The frame coherence constraint tries to maintain a level of coherency in the distance the camera moves in the current frame in relation to that in the previous frame. This constraint acts as an acceleration or deceleration force. Frame coherence and visibility constraints are assigned weights, but they have no explicit desired values. Occlusion constraint is not used until there is an occlusion problem, in which case it influences how quickly the camera moves away from the occluded positions. They claim that this method has the advantage of avoiding contact between camera and scene

geometry; explicit collision detection between camera and scene geometry and its response is rarely necessary. They also claim that this integration of visibility maintenance method with the constraint solver enables it to become applicable to any environment and domain.

A constraint directed stochastic local search algorithm is used to search the domain of constraint values. The solver searches iteratively, and evaluates each potential solution and keeps the best one. After a specified number of moves the best one is returned as the solution. They suggest using a problem specific search heuristic to take advantage of the nature of the problem such as ordered domain, and geometric nature of the problem. As an example they propose the heuristic of what they call "competitive strategy" to prune the domain. In this strategy the constraint with the highest cost (dominant constraint) attains temporary control of the search and prunes the search space where its cost of violation is higher.

The set of constraint values and their respective weights is called the camera profile. Different camera profiles generate different types of camera behaviour. The behaviour of the camera can be changed by changing the camera profile at run time with the use of simple interpolation of the profiles during the transition.

They claim that their camera system performs all of the visualization requirements of existing works, including Bares, McDermott et al. (2000), Bares, Thainimit and McDermott (2000) and Halper et al. (2001), and that it is better than those systems in terms of cost. They also claim that their representation has reduced the problem of camera control into selecting an effective and efficient constraint solver, and that it can be

extended to include new constraints without modification of representation or constraint solver.

Bourne and Sattar (2005b) further extend their above idea. They address the problem of automatically generating constraint weights appropriate for the properties, and thus relieving the artists and game designers of determining them using a trial-and-error process. They use a genetic algorithm which takes a set of example animation traces as input and produces the corresponding constraint weights by searching the space of possible camera profiles.

The above method is further extended in Bourne (2006) and Bourne and Sattar (2006). They analyze the cost structure of the domain of search of the constraints such as distance, height and orientation (for a tracking shot) for different weights and, using the result, they propose a sliding octree solver which takes advantage of that information to search the domain efficiently (details are given in section 2.6.2).

Jardillier and Languenou (1998) and Languenou et al. (1998) use the pure interval method to calculate the whole set of camera movements in virtual environment satisfying user constraints on screen and/or on camera. They assume that the objects' behaviour is completely known in advance by the system. The camera path is assumed as a parameterized function of degree 3 for each degree of freedom of the camera. Time is also considered as a constraint. The unknowns are the parameters of the function. They use interval arithmetic for the constraint solver which is based on recursive subdivision of the search domain. They claim that there is no key framing and no interpolation and hence the solutions are guaranteed to satisfy the constraints. The authors say that they

consider only simple movements, and do not consider occlusion avoidance or collision detection.

The interval arithmetic based approach is computationally expensive, especially for many variables and large domains, but it has the advantage of providing guaranteed approximation of all the solutions, or return false if there is no solution (Christie et al., 2002; Christie and Olivier, 2006).

Christie et al. (2002) and Christie and Languenou (2003) improve the above interval based method by reducing the number of variables and directing the search with propagation of good canonical solutions. They also address the problem of occlusion avoidance and collision detection. They represent the path of the camera as a constrained combination of primitive camera movements based on cinematography that are sequentially linked together. The primitives include panning, traveling, arcing and any combination of these movements. Each primitive camera movement, called hypertube, is treated as a separate constraint satisfaction problem. The constraint solver uses depth first search and during backtracking it uses tabu strategy. The hypertubes are solved in sequence ensuring the end of the i^{th} hypertube joins the beginning of $(i+1)^{th}$ hypertube.

Christie and Olivier (2006) claim that in the interval based approach the fulfillment of properties for the whole sequence is guaranteed. They also argue that the main problem with the interval based approaches is that when there is no solution the solver exits without any information about the inconsistencies. This then requires the user to remove some constraints and run the process again iteratively until a solution is found.

2.8 Conclusion

This chapter provides the underlying concepts of automatic camera control in the virtual environment in general, and computer games in particular. An overview of the constraint satisfaction techniques that have been proposed to date for solving the problem has been given. The next chapter describes our proposed method to solve the problem.

Chapter 3

The Camera Control System

Our motivation for this research (Chapter 1) and its current state (Chapter 2) have been described in the previous chapters. In this chapter we present an analysis of the representation of the camera control problem using weighted constraint, and propose a solution for the problem on the basis of this analysis.

3.1 Problem Formulation

In this thesis we shall consider a simple tracking shot of a single target. The most important visual properties are represented by the size of the subject's image in relation to the frame and viewpoint (Katz, 1991; Mascelli, 1998). The viewpoint is determined by the orientation and the camera height or the angle of view. The image size is determined by the distance of the camera from the subject and the focal length of the camera. So, we can use a relatively larger range for the domain of distance and adjust the focal length after determining the most appropriate distance. The visual requirements of the distance and orientation can be expressed as constraints on the distance and azimuth angle of the polar coordinate system whose pole is at the position of the subject in the current frame and polar axis is along the horizontal projection of the direction of line of action (e.g.,

along the direction of facing or the direction of movement of the subject). We shall use the zenith angle of a spherical polar coordinate system to represent the angle of view constraint. The positive z-axis of the coordinate system is along the vertical line. Then the constraint will be independent of the distance and orientation constraints. Whatever may be the variations in the distance and orientation, the shot will always be at the same camera angle, such as at eye level, as long as the zenith angle is within its acceptable range of values. But, if we use the height coordinate of a cylindrical coordinate system to represent this constraint, for some values of the distance within its range of acceptable values, the shot may not be at eye level, it may become a high angle or low angle shot. Let the desired position of the camera be specified as a point with coordinates $(\rho_d, \theta_d, \varphi_d)$ with respect to this spherical coordinate system (ρ, θ, φ).

If the potential position of the camera is at $(\rho_p, \theta_p, \varphi_p)$, the costs for the visual constraints are given by:

$$\rho_1 = | \rho_p - \rho_d |$$

$$\theta_1 = | \theta_p - \theta_d |$$

$$\varphi_1 = | \varphi_p - \varphi_d |$$

If k_1, l_1 and m_1 are the corresponding weights, then the weighted cost for the visual constraints is:

$$k_1\rho_1 + l_1\theta_1 + m_1\varphi_1$$

Similarly, we use $(\rho', \theta', \varphi')$ as the spherical coordinate system for the motion constraints of the camera with the pole at the position of the camera in the previous frame, horizontal projection of the direction of movement of the camera in the previous frame as the polar axis and the vertical line at the pole as the positive z-axis. Let the desired position of the camera according to the motion constraints be $(\rho'_d, \theta'_d, \varphi'_d)$. If the potential position of the camera is at $(\rho'_p, \theta'_p, \varphi'_p)$ with respect to this coordinate system, then the costs for motion constraints are given by:

$$\rho_2 = | \rho'_p - \rho'_d |$$

$$\theta_2 = | \theta'_p - \theta'_d |$$

$$\varphi_2 = | \varphi'_p - \varphi'_d |$$

If k_2, l_2 and m_2 are the corresponding weights, then the weighted cost for the motion constraints is:

$$k_2\rho_2 + l_2\theta_2 + m_2\varphi_2$$

The total weighted cost for the problem is given by:

$$k_1\rho_1 + l_1\theta_1 + m_1\varphi_1 + k_2\rho_2 + l_2\theta_2 + m_2\varphi_2 \qquad (3.1)$$

3.2 Determination of the Solution

First we consider a 2-dimensional problem and then extend it to 3 dimensions. We first consider visual constraints of distance and orientation relative to a single target. Let the total weighted cost is

$$k_1\rho_1 + l_1\theta_1$$

where k_1 and l_1 are the constant weights.

The system of isocurves Γ_1 of visual constraints (Figure 3.1 shows a member γ_1 of the family Γ_1) are given by

$$k_1\rho_1 + l_1\theta_1 = c$$

where c is a constant. We note that the cost on an isocurve is proportional to the distance cost when $\theta_1 = 0$. It is also proportional to the orientation cost when $\rho_1 = 0$. Let ρ_{10} and θ_{10} be the ranges of acceptable values for ρ_1 and θ_1 respectively. Since A and B are acceptable solutions lying at the edges of the two constraint ranges keeping the other constraint's cost to zero, they must have the same total cost. Thus, the curve ABCD has total constant cost, and its equation is given by

$$\rho_1/\rho_{10} + \theta_1/\theta_{10} = 1$$

and the weights for distance and orientation are inversely proportional to ρ_{10} and θ_{10} respectively. Thus, the isocurves of total constant cost are given by

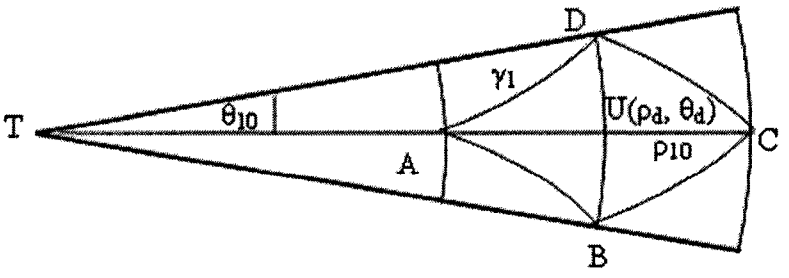

Figure 3.1: An isocurve γ_1 of visual constraints.

$$\rho_1/\rho_{10} + \theta_1/\theta_{10} = \text{constant}$$

If the total cost is greater than 1 the potential position for the camera is outside the domain of visual constraints and hence of less than acceptable quality. Here the constraint costs can be viewed as normalized costs and the weights are equal, viz., 1.0.

Similarly, we can find the isocurves and the weights for the linear and angular frame coherence constraints from their desired values and the ranges of acceptable values. The results will be similar. The weights here will act as linear and angular acceleration / deceleration.

The following theorem reveals the cost structure of the domain for the visual and the motion constraints. We prove the theorem for visual distance and orientation constraints. It also holds for frame coherence distance and rotation constraints.

Theorem 3.1: Isocurves of less cost are contained within isocurves of higher cost.

Proof: Let γ_1 be an isocurve (Figure 3.2) with total cost c given by

$$\rho_1/\rho_{10} + \theta_1/\theta_{10} = c$$

Let P be a point inside γ_1. Let TP meets with γ_1 at R and intersects with BD at Q. Then R has the total constraint cost of c. Since P is inside γ_1, $QP < QR = \rho_1$. Since the points P and R have the same angle θ_1, the total cost at P is less than that at R. So, any point inside γ_1 has less cost than that on γ_1. Similarly, we can show that any point outside γ_1 has higher cost than that on γ_1.

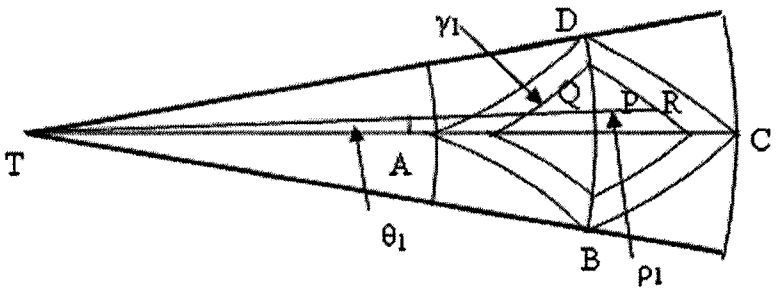

Figure 3.2: Isocurves of less cost are contained within isocurves of higher cost.

Hence all the points with cost less than that of γ_1 are contained inside γ_1. This proves that each curve passing through points having total cost less than that of γ_1 lies inside γ_1.□

Now, let the total weighted cost of visual and frame coherence constraints for a potential position of the camera be c. From (3.1) we have

$$k_1\rho_1 + l_1\theta_1 + k_2\rho_2 + l_2\theta_2 = c \tag{3.2}$$

We have to find the point where c is the minimum.

Frame coherence is necessary to keep a viewer's attention to the action in the image and to nothing else. Without it, the viewer's attention will be drawn to the camera (Thompson, 1998). On the other hand, adequate coverage is the minimum requirement for the visual effect of a scene (Katz, 1991). So, frame coherence must be given higher priority than the visual constraints. Frame coherence constraint cannot be relaxed beyond its acceptable range, but visual constraint can be relaxed as much as the situation demands provided there is adequate coverage. Consequently, frame coherence constraint will limit the search domain to promising regions of space. Within this region the frame

coherence constraint can be relaxed more readily than other constraints. So, once we identify this region we can ignore frame coherence constraints. Also the nature of their effect on the cost potential is different from that of the visual constraints. So, we need to decouple the cost potential for frame coherence constraints from that for visual constraints to analyze the cost structure of the search space and take advantage of that to control the camera in an informed way.

To that end we decompose the above problem (3.2) into two – one for the frame coherence constraints and the other for the visual constraints:

$$k_1\rho_1 + l_1\theta_1 = c_1$$

$$k_2\rho_2 + l_2\theta_2 = c_2$$

where $c_1 + c_2 = c$. We have to find c_1 and c_2 such that c is the minimum. These two equations define two systems of isocurves for their respective constraints. All the points on a particular isocurve have the same cost with respect to the cost potential of the isocurve given on the left hand side of their respective equation.

Now, we can specify the visual and the motion weights separately by considering the acceptable domains of visual and motion constraints respectively. The weights are determined automatically once we identify the respective acceptable domains. Moreover, the solution will always be within their common domain – if they have an acceptable common region. Thus, one need not consider the weights. Only the most appropriate desired positions and range of acceptable positions of camera with respect to visual constraints and frame coherence constraints need to be determined.

The points of intersection of the two families of isocurves will have the cost that is the total of the costs of the two isocurves. So, if we find the point of intersection of the two systems of isocurves that has the total minimum cost, that point will be the solution for the total problem. Obviously the locus of the point of contact of the two families of isocurves will contain the minimum cost point. The following theorem helps us find this point.

Theorem 3.2: If within a certain region of space one of the visual or motion constraints has higher priority, then the total least cost for all of the visual and motion constraints will be at the end point of the locus of the point of contact within that region of the two systems of isocurves for visual and motion constraints that has the lowest cost for the higher priority constraints.

Proof: Let the total weighted cost be given by (3.1). The points with constant weighted cost are given by

$$m_1(\rho_1 + n_1\theta_1) + \rho_2 + n_2\theta_2 = c$$

where c is a constant for the particular locus of point of total constant cost. The systems of isocurves Γ_1 and Γ_2 with constant cost for visual and motion constraints respectively are given by:

$$m_1(\rho_1 + n_1\theta_1) = c_1 \tag{3.3}$$

$$\rho_2 + n_2\theta_2 = c_2 \tag{3.4}$$

where $c_1 + c_2 = c$.

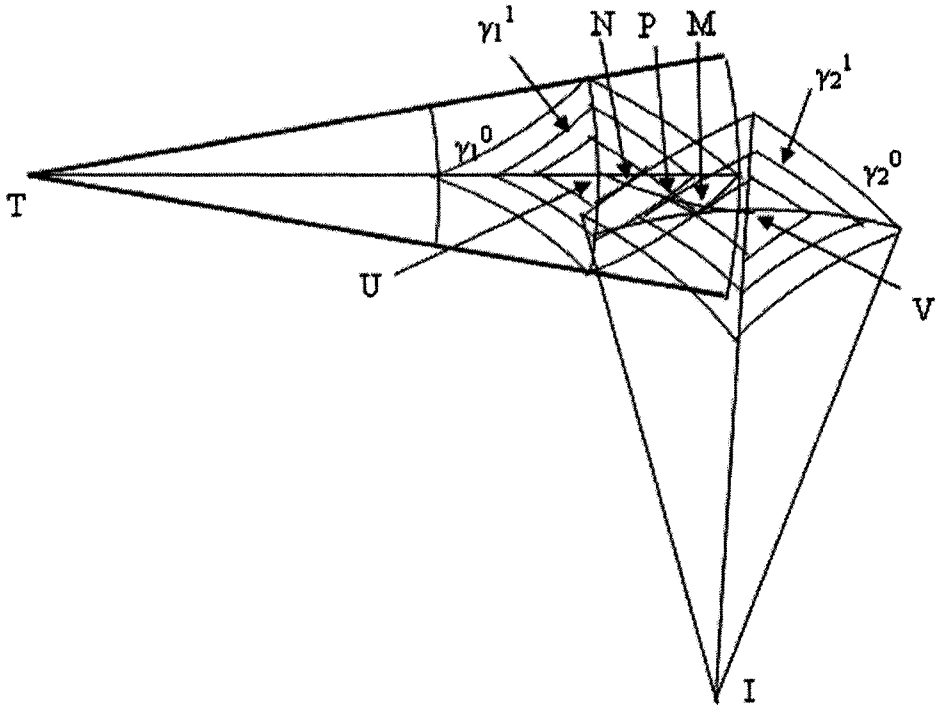

Figure 3.3: Point of contact of the isocurves of visual and motion constraints.

Let P be the point of contact of the isocurve γ_2^1 of motion constraints with isocurve γ_1^1 of visual constraints and UPV be the locus of the point of contact (Figure 3.3). The other curves from Γ_1 intersecting with γ_2^1 will contain P and hence by Theorem 3.1 they will have more visual cost than that on γ_1^1. Since all the points of γ_2^1 have the same motion cost, those other intersecting points will have more total cost for combined visual and motion constraints than that at P.

Let the isocurves γ_1^1 and γ_2^1 intersect with their respective axis at ρ'_1 and ρ'_2 respectively. Then, from (3.3) and (3.4) we see that the cost for their respective isocurves will be $m_1\rho'_1$ and ρ'_2 respectively. So, the total cost at the point of contact will be

$$m_1\rho'_1 + \rho'_2 \tag{3.5}$$

Successive interior curves of one family will be in contact with the successive exterior curves of the other family. So, if the costs $m_1\rho'_1$ of successive interior curves of Γ_1 decrease, the costs ρ'_2 of the corresponding tangential successive exterior curves of Γ_2 increase, and similarly the other way around.

So, in the total cost expression given by (3.5), if ρ'_1 increases then ρ'_2 decreases, and vice versa. Since (3.5) is linear in ρ'_1 and ρ'_2, and since ρ'_1 and ρ'_2 are non-negative and bounded, we can select m_1 sufficiently large within a certain region bounded by the isocurve $\gamma_2{}^0$ of motion constraints to make the visual constraints higher priority than the motion constraints within that region which will make the total cost in (3.5) minimum when ρ'_1 is the minimum. So, the minimum cost point will be at an end point N of the locus VMPN of the point of contact within that region that has the lowest cost for the visual constraints.

Similarly, by making m_1 sufficiently small we can make the motion constraints higher priority than the visual constraints within a certain region bounded by the isocurve $\gamma_1{}^0$ of visual constraints, for which case the minimum cost solution will be at the end point M of the locus UNPM of the point of contact within that region that has the lowest cost for the motion constraints. Similarly, it can be shown that the theorem also holds for the other alignment of the isocurves of visual and motion constraints.□

In Figure 3.3 suppose we can relax the motion constraints readily within a certain region bounded by the isocurve $\gamma_2{}^1$ of motion constraints. Then Theorem 3.2 shows that the minimum cost solution will be at P. We can use this property of the constraints to better control the motion or the visual effect of the camera. For example, we can move the

camera most vigorously (of course, frame coherently) by relaxing the motion constraints for the whole domain of frame coherent motion. For this case the visual quality will be the maximum for the problem. To further increase the visual quality we can even further increase the region of relaxation of the motion constraints. But that will make the camera motion unsmooth. Or, we can relax the motion constraints for only a part of its maximum acceptable range of relaxation to move the camera very smoothly.

In the extreme cases, if the motion constraints cannot be relaxed at all (i.e., if it has more priority than the visual constraints throughout the whole domain) then the solution will be at the desired position of the camera according to the motion constraints and the camera will be moving at the same speed in the same direction. On the other hand, if the visual constraint cannot be relaxed at all then the camera solution point will be at the desired position of the camera according to the visual constraints, and the camera will be moving erratically always going to the best possible viewing position.

If only one of the motion constraints, say linear motion of the camera, can be relaxed and the other can not be relaxed at all then the camera will be moving in the same direction with variable speed. The solution camera position will be at any point on the straight line along the motion direction within the range of acceptable values for linear motion. Similar will be the case if only the rotation speed of the camera can be relaxed but its linear speed can not be relaxed.

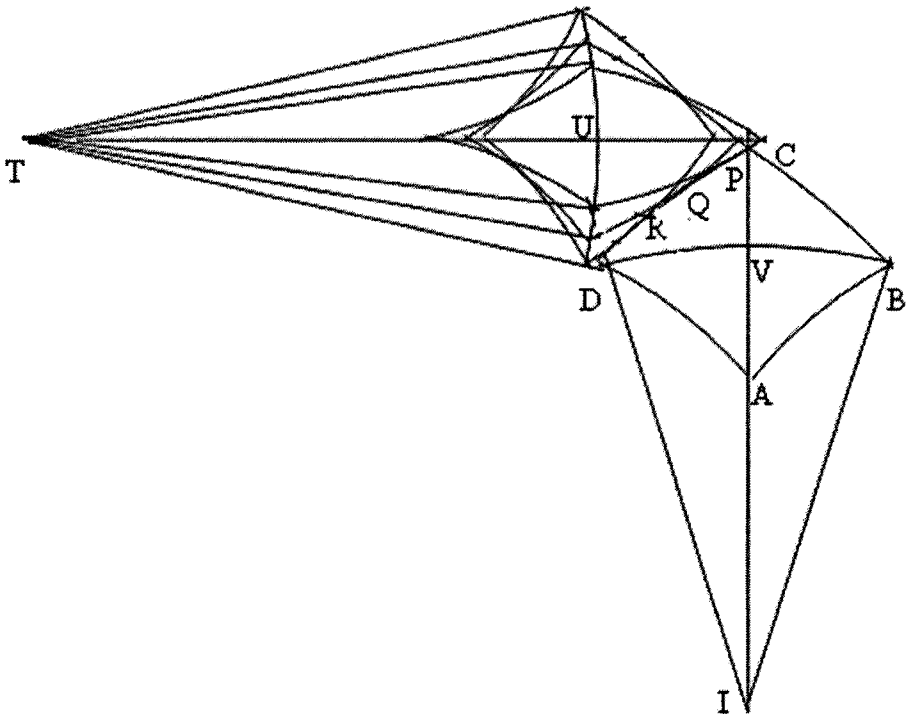

Figure 3.4: Effect of constraint weights.

3.3 Effect of Weights

The combined effect of the constraint weights on the motion of the camera is shown in Figure 3.4. In this figure we use higher priority for the visual constraints within the region bounded by the isocurve ABCD of motion constraints.

Figure 3.4 shows that for the values of frame coherence weights such that its isocurve has axes of equal length, the weights for visual distance and orientation constraints that have isocurve of equal length axes will attract the camera solution point toward the desired position U according to the visual constraints in a uniform manner (the solution point is Q in Figure 3.4). Higher weight for the visual distance constraint attracts the camera more rapidly towards the positions of desired distance than the positions of

desired orientation (the solution point is R in Figure 3.4), and higher weight for the visual orientation constraint attracts the camera more rapidly towards the positions of desired orientation than the positions of desired distance (the solution point is P in Figure 3.4). Similarly, we can show the total effect of weights for the case of higher and lower weights for frame coherence distance than frame coherence rotation.

The case will be similar for giving higher priority to the motion constraints within a region bounded by an isocurve of visual constraints. In this case the role of those two groups of constraints will be interchanged.

3.4 Control of Weights at Frame Level

In Bares, McDermott et al. (2000), Bares, Thainimit and McDermott (2000), Bourne and Sattar (2005a, 2006) and Bourne (2006) constraint weights are determined and applied at the level of a simple shot (i.e., for the whole length of a simple shot). They cannot be determined or specified at the frame level. During the transition the camera profiles are interpolated by Bourne and Sattar (2005a, 2006) and Bourne (2006). In our approach, weights can be determined and applied at all levels including simple shot level and frame level. Application of weights at frame level is necessary if some constraints are affecting only portions of a simple shot.

Hierarchical control of weights is necessary to have finer control on the visual effect or camera movement. Depending on the situation they are controlled down to the frame level. Frame level weights will have the highest precedence, developing shot level weights will have the lowest precedence, and the simple shot level weights will have the

precedence in between. For example, at the simple shot level if we can afford to have less control for visual quality within a region, we can apply stricter frame coherence weight within that region and the camera motion becomes very smooth. For portions of the simple shot it may be necessary to increase the frame coherence domain to its maximum possible extent to have a common domain to position the camera, thus reducing the weights for frame coherence constraints at the frame level.

Once new desired values and weights are assigned to the constraints, the camera will automatically be guided toward the desired position smoothly. No interpolation is necessary. The camera may be accelerated or decelerated radially or angularly at the frame level by higher level adjustment of the weights of camera motion and the relative priority of the motion constraints with respect to the visual constraints.

3.5 Strategy for Using the Constraints

The effect of all the requirements and hence the constraints of a shot on camera parameters and its motion are not similar. To control the camera intelligently we need to have prior information about the effect of each constraint and the resultant effect of all the constraints before their application. Since decomposing the problem helps us to know and precisely control the effect of each constraint in the overall problem involving all the constraints of the camera, it is desirable to classify the constraints as visual constraints and motion constraints. Their combined effect will determine the position of the camera. The centre of view and view up vector determine other three parameters of the camera. Finally, determining the field of view fixes the focal length. Thus all the seven parameters of the camera are determined.

For that we need to group the constraints in relation to the type of camera parameters or camera motion they are interacting with. The classification is given below:

- Camera Motion Constraints: They are related to the frame coherent motion of the camera. They include frame coherent constraints, and all other constraints related to the movement of the camera, viz., slow or fast moving camera, jerky camera, ascending or descending camera, tracking camera, etc. This group will also include the constraints that will guide the camera to move to a desired region of space in the future frames by using prediction to avoid collision of the camera with the environment elements, or to avoid occlusion of, say, dramatic circle of interest by environment elements, or to transition to another shot within the same developing shot. Each of them will have the most appropriate value and a domain of acceptable values, the range of which determines the weight for it.
- Visual Constraints: This group includes all other constraints. All these constraints are related to the quality of the image. They include, for example, distance, orientation, camera height, depth order, etc.
- Centre of View Constraints: They include location of subject / subjects of the shot on the image, object inclusion constraint, etc.
- Field of View Constraints: This group consists of such constraints as dramatic circle of interest, shot size on the image, object inclusion constraint, etc.

Some of the constraints may be hard constraints, e.g., frame coherence constraints, and avoiding occlusion of eyes in extreme close up. They must be satisfied. Most of the constraints are soft constraints. Each of the soft constraints in each group are satisfied in

the best possible way with respect to other constraints in their respective groups using the weighted constraint method or any other method appropriate for that particular constraint. In this way, camera motion constraints and visual constraints will produce two acceptable domains with the most appropriate desired position in their respective centres. The position having the least total cost for these two groups of constraints will be the position of the camera.

Once the camera is positioned there, the centre of view is determined by considering the related constraints. Finally, the field of view is adjusted by using its related constraints. These two are also adjusted smoothly. The strategy described here is very similar to the real cameraman as he moves the camera smoothly to the best possible position and adjusts the centre of view and field of view accordingly.

3.6 Strategy for Using the Weights

Knowing the behaviour of the camera position (corresponding to visual constraints) and motion (corresponding to motion constraints) in relation to their respective weights the camera module can control the camera in an informed way. Different strategies can be used for different types of shots to determine the appropriate weights for that type of shot. An example strategy would be to use a very restrictive domain (may be a one point domain in the extreme case) for the camera motion constraints in the first attempt (Table 3.1). More weight is given to visual constraints so that the solution lies on the outer isocurve of the restrictive domain of camera movement.

	Motion Constraints	Visual Constraints
1st attempt	Most restrictive domain	Most restrictive domain More weight
2nd attempt	Relax domain More weight	Domain is as in 1st attempt
3rd attempt	Domain is as in 2nd attempt	Relax domain More weight
Final attempt	Domain is as in 3rd attempt	Relax the domain to its maximum possible extent More weight

Table 3.1: Using the weights

If there is no common region in the first attempt, the camera motion domain can be relaxed, but more weight can be given on it so that the solution lies within its domain. This can be achieved by searching along the bounding isocurve of visual constraints that bounds the acceptable region of visual constraints. If the second attempt fails, the domain for the visual constraints can be relaxed and the visual constraints can be given higher priority than the motion constraints. If the third attempt fails but still then we need a solution, the domain for the visual constraints can be relaxed to its maximum extent and the higher weight is maintained. The visual quality of the image may be very poor. This option can be used in such cases as during the computation of the next shot for which it is not possible to cut to another shot in the current frame.

This hierarchy of decision is based on the relative importance of frame coherent movement of the camera versus the visual effect of the scene. The above hierarchy may be modified if the shot demands differently.

3.7 Solving for Camera Position

We consider only one scenario where the visual constraints have higher preference (i.e., weights) than the motion constraints within the domain of motion constraints and lower preference outside that. We also assume that all the weights are constant. So, both the systems of isocurves will be either convex or concave in each quadrant. According to Theorem 3.2, if the desired position of the camera according to the visual constraints is inside the domain of the motion constraints then the solution camera position is at that desired position, otherwise the solution camera position will be at one of the points of contact of the outermost isocurve for motion constraints with an isocurve for visual constraints where the cost of the visual constraints is the minimum.

But there is no known exact solution. So, we need to search the outermost isocurve for motion constraints to find the point where the total weighted cost of the visual constraints is the minimum. We shall use a binary search in which the search domain is successively refined into one half of its previous size and the search continues in the region which is known to contain the point with the minimum total weighted cost for the visual constraints.

First we consider the search of an arc such as a quadrant AB, or its part, of the isocurve of motion constraints (Figures 3.5a, 3.5b, 3.5c and 3.5d). On the arc AB of Figure 3.5 there may be the only minimum for the arc (absolute minimum in Figure 3.5a and local minimum in 3.5b), or a minimum at each of the two end points and one maximum in between (Figure 3.5c), or one minimum and one maximum at the end points (Figure 3.5d).

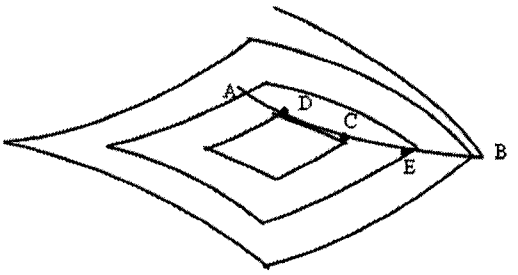

Figure 3.5a: Quadrant AB contains the point with the absolute minimum cost for visual constraints.

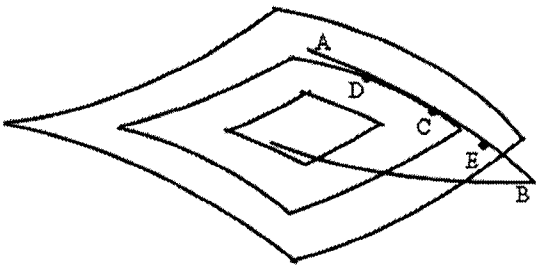

Figure 3.5b: Quadrant AB contains the point with local minimum cost for visual constraints.

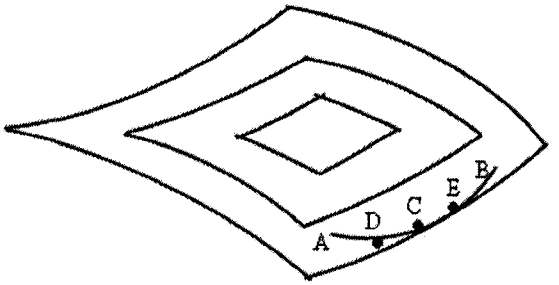

Figure 3.5c: Quadrant AB contains the point of contact with a maximum in CEB. Local minima at A and B.

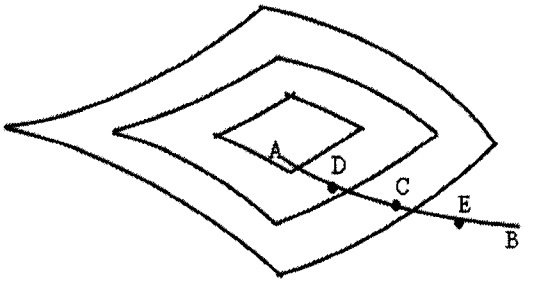

Figure 3.5d: Quadrant AB has no point of contact. Local minimum and maximum at A and B respectively.

Figure 3.5: Searching a quadrant or a portion of it.

To search, say, AB we first evaluate the cost for visual constraints at A and B. Then we divide AB by the point C and calculate the cost for visual constraints at C. Then we divide the two halves AC and CB of AB by D and E respectively. We note that the quadrant AB contains either one minimum or two minima for the quadrant. The absolute minimum for the quadrant is either at the point of contact of AB with an isocurve of the visual constraints or at one of its end points. The two minima are at the end points A and B. We check the arcs ADC, DCE and CEA to find the one that contains the point of contact having the absolute minimum cost for the quadrant. If they do not contain the point of contact with the absolute minimum cost then the arcs AD or EB contains the point of contact or the end point having the absolute minimum cost for the quadrant. The algorithm is given below:

If cost(D) <= cost(A) and cost(D) <= cost(C)

Then ADC contains the minimum cost position

Else if cost(C) <= cost(D) and cost(C) <= cost(E)

Then DCE contains the minimum cost position

Else if cost(E) <= cost(C) and cost(E) <= cost(B)

Then CEB contains the minimum cost position

Else if cost(A) is the minimum among these five points

Then AD contains the minimum cost position

Else EB contains the minimum cost position

If the resultant arc was only a quarter of the arc in the previous iteration (AD in Figures 3.5c and 3.5d) we follow the procedure of searching AB. If the resultant arc is one half of the arc in the previous iteration then it contains the point of contact with the absolute minimum for the quadrant (ADC in Figures 3.5a and 3.5b). We divide the two halves of this resultant arc and continue our search to the portion of the arc that contains the point of contact. The length of this portion of the arc will be one half of the length of the arc in the second iteration. We continue until the search domain is refined to the desired accuracy. This method has the advantage that the search domain is partitioned into increasingly smaller regions with decreasing cost. For all the cases of Figure 3.5, the absolute minimum point for the arc AB is found by this search.

In each iteration the resultant arc length reduces by one half, and 2 mid-points are evaluated, except for the first iteration. In the first iteration 3 points are evaluated. So, to refine the domain size to $(1/2^n)^{th}$ we need to search $(2n + 1)$ points. In particular, to refine

the domain size to < 1%, we need to reach $(1/2^7)^{th}$ of the domain size. This will require only 15 points to search.

Now we consider the whole isocurve of motion constraints for searching. We see from Figure 3.6 that there will be more than one point of contact. In this figure there are two local minima (at Q and R) and one absolute minimum (at P). In addition, there is one maximum (at C). For all the cases the binary search algorithm will find the absolute minimum point for each of the quadrants. The algorithm can be applied to all the four quadrants separately and, among the resulting four minimum points, the absolute minimum point will be the solution point. To reduce the search space to $(1/2^7)^{th}$ each quadrant should be divided five times. The algorithm needs to search a total of 4 x (2 x 5) or 40 points.

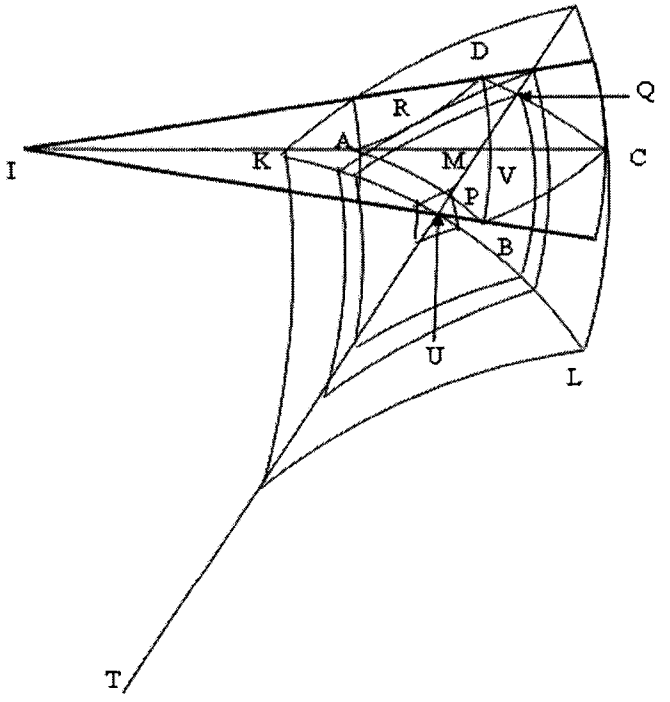

Figure 3.6: Absolute minimum (P), local minima (Q) and local maxima (R and S) on the same isocurve.

To improve the performance we avoid searching the quadrants that are known not to contain the absolute minimum for the whole isocurve of motion constraints. In this thesis we consider only one situation corresponding to a particular relative position of T and U with respect to IV such as when they are below IV (Figure 3.6), and describe how to find the quadrant(s) on which we have to apply the binary search. First we consider if the angle TMI is an acute angle:

If TU intersects with IV on the left side of A:	search DAB
Else if TU intersects with AB:	search AB
Else if TU intersects with BC:	
If KL intersects with BC:	search BC
Else:	search ABC
Else:	
If TU <=TOB:	search ABC
Else if TB <= TU <= TC:	search BC
Else:	search BCD
If the angle TMI is greater than or equal to a right angle:	
If TU intersects with IV on the right side of C:	search BCD
Else if TU intersects with BC:	search BC
Else if TU intersects with AB:	
If KL intersects with AB:	search AB
Else: (i.e., if KL intersects with BC)	search ABC
Else: (i.e., if TU intersects with IV on the left side of A)	
If TU <= TB:	search ABC

Else if TB <= TU <= TA:	search AB
Else: (i.e., if KL intersects with AD)	search DAB

We note that at most two adjacent quadrants are selected. In the above pruning process, if TU or KL intersects twice in the same quadrant of ABCD, we can eliminate that part of the quadrant that is farther from U. Similarly, for the other three situations corresponding to other relative positions of T and U we can use the properties of the two systems of isocurves to remove the quadrants that are known not to contain the absolute minimum, and select one or two adjacent quadrant(s) or their parts that contain the absolute minimum. If one quadrant or its part is selected we search it using the binary search method.

If two quadrants or their part are selected one of them will contain the absolute minimum and the other will contain local minimum only at the common point of the two octants or of their parts. To find the quadrant or the portion of the quadrant that contains the absolute minimum we evaluate the common point of the two quadrants and its adjacent points on both sides of it at the granularity of the refinement of the problem. If the common point has the minimum cost then one of the quadrant has only one minimum at that point and the other quadrant or its part has minima at its two end points, i.e., at the common point for the two quadrants and at the other end of that arc. The minimum between these two points is the solution point.

On the other hand, if the common point for the two quadrants does not have the minimum cost the quadrant or the portion of the quadrant that contains the minimum cost point among the three points will contain the absolute minimum point. We search this

quadrant or its part using the binary search method. This will require 10 more points to be evaluated to reach the refinement of $(1/2^5)^{th}$ of the quadrant, i.e., $(1/2^7)^{th}$ of the whole isocurve. Adding the common point and its two adjacent points the total number of points to be evaluated is 13.

One advantage of this algorithm is that we follow the isocurves of successive lower cost in each iteration to reach the minimum cost solution. We can use an ordered set of numbers to track these areas of cost hierarchy where the first element corresponds to the first pass, second element corresponds to the second pass, and so on. The serial number of only the minimum cost point in a pass is stored in its corresponding element.

Since there are five points to compare in each pass along a curve we use numbers from 0 to 4 to represent the position of the minimum cost point in a pass. So, each element in the set of ordered numbers will be from 0 to 4. The five points for each pass are numbered from 0 to 4 in one common direction along the isocurve of search. The first element of the ordered set of number contains the serial number of the minimum cost point in the first pass, the second element contains the serial number of the minimum cost point in the second pass, and similarly for the other elements. The last element will give the serial number of the solution point.

For the special cases when the minimum cost point is at one of the edge points of the current interval (recognized by the minimum cost point number 0 or 4 in the corresponding element for the pass in the ordered set of numbers) the search confines to the two point subinterval at the corresponding edge which has the size of $(1/2^2)^{th}$ of the current interval of search. We use only one mid-point for the next pass, but the search

interval for the following pass is not reduced by this. So, the next element in the ordered set will contain 0 to 2. The next pass will again have five points to compare. In the first iteration the search domain is divided into $(1/2^2)^{th}$, so the number of elements in the ordered set of numbers will be one less than the number of passes. If we refine a search space into $(1/2^5)^{th}$, the ordered set would have four elements.

In Figure 3.7 an example search with its related ordered set and minimum cost points in each pass are shown. In the first iteration the search domain is divided into $(1/2^2)^{th}$, so the number of elements in the ordered set of numbers will be one less than the number of passes. Thus if we refine a search space into $(1/2^5)^{th}$, the ordered set would have four elements.

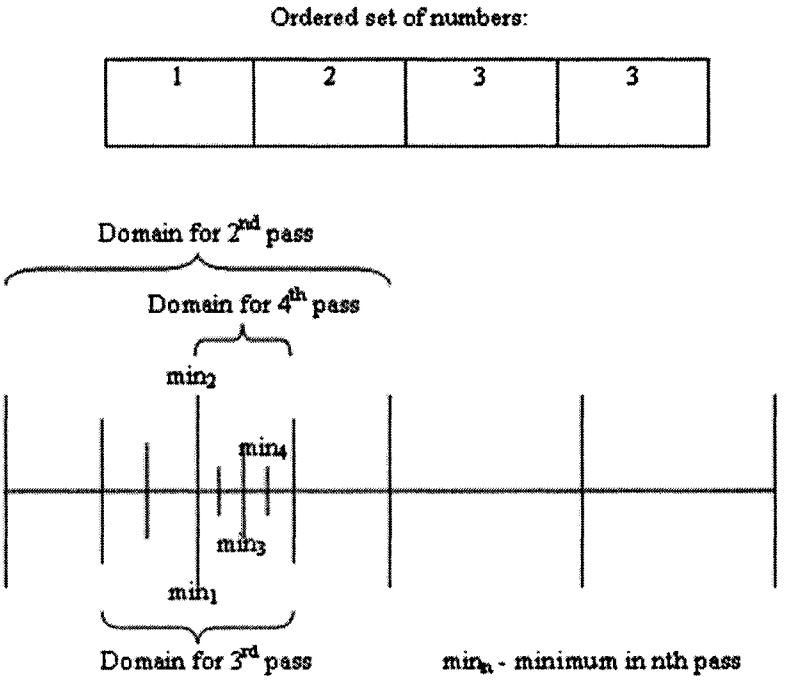

Figure 3.7: Ordered set of numbers representing the search domain hierarchically.

We note that each pass compares five points to find the minimum among them and confines the search to the half of its domain containing three of its points in the next pass. If for any reason, e.g., the problem of occlusion, the minimum cost solution is not a valid solution, we check the four other points corresponding to the last pass for a valid solution. They cover half of the domain size in the previous pass. If all these five points are invalid, we need to check the other half of the domain in the previous pass. This can be done by checking the other two points of the previous pass and searching that half domain down to the level of refinement for the problem. If that half is also invalid, then all the domain of this pass is invalid. This is one half of the domain of its previous pass. We move on to the previous pass and search the other half of its domain down to the level of refinement for the problem. We continue this process of moving on to the next higher pass and checking for validity down to the level of refinement for the problem. The process continues until all the passes including the first pass are refined and checked for validity. If it fails, there is no valid point for the camera position and all the points have been checked. But we do not need to check the whole domain. We can use information about the occluded areas and the ordered set of numbers about the cost hierarchy to avoid the occluded areas and jump to the immediate next higher cost area that is unoccluded.

If the minimum point is found directly at a point of intersection without searching we can use the reverse process of binary search to expand the search domain in each iteration to double of its previous size with with the new half having the next best cost point, and check that new half for occlusion. If that half is also occluded the search domain is expanded again to double of its present size. But if the new half has an

unoccluded area then that is searched and checked for occlusion using binary search as before down to the level of refinement of the problem.

3.8 Extension to 3D

Let the total weighted cost for the 3D problem be c. Then we have

$$k_1\rho_1 + l_1\theta_1 + m_1\varphi_1 + k_2\rho_2 + l_2\theta_2 + m_2\varphi_2 = c \tag{3.6}$$

With the straightforward extension of our approach to 3D, we decompose (3.6) into two separate problems for visual and motion constraints respectively:

$$k_1\rho_1 + l_1\theta_1 + m_1\varphi_1 = c_1$$

$$k_2\rho_2 + l_2\theta_2 + m_2\varphi_2 = c_2$$

where $c = c_1 + c_2$. These two equations define two systems of isosurfaces. A straightforward extension of Theorems 3.1 and 3.2 to 3D readily apply to them. Again, we consider the visual constraints having higher preference than the motion constraints within a certain region bounded by an isosurface γ_2 of motion constraints, and that all the constraint weights are constant for the current frame. Similarly to 2D, if the desired position of the camera according to visual constraints is not within the region bounded by the isosurface γ_2 of motion constraints, then the solution camera position will be at the point of contact of γ_2 with an isosurface of visual constraints, and we search on γ_2 to find the point where the total cost of the visual constraints is the minimum. We observe that the isosurface of motion constraints will intersect with the isosurface of visual constraints at isocurve of total cost of visual and motion constraints, and that these isocurves will be

inclusive of one another, i.e., each will be contained within the others of higher total cost. We utilize this property and direct our search successively towards the inner curves to reach the minimum cost position.

We shall use an extension of the binary search algorithm used for 2D. We use the technique of binary slice and search where we slice the search area successively into two halves and confine our search into the area that is known to contain the inner isocurves of total cost of visual and motion constraints completely. This will ensure that the refined search area contains the solution point. The refined search area in each pass is of the size of one half of the size of the search area in the previous pass.

First we describe a brute force search method. Again, let us consider the search of a portion, such as an octant AOB, of the isosurface of motion constraints (Figure 3.8). To begin, we search the arcs OA, OB and the middle arc OC using our binary search method for 2D up to the desired level of refinement of the problem (in Figure 3.8 we have shown the subdivision five times along the arc OB only). This will find the minimum points on the curves up to the required level of refinement. Then we search the middle arcs OD and OE of the two halves AOC and COB using our binary search method for 2D up to the desired level of refinement of the problem. Among the arcs OA, OB, OC, OD and OE we identify the one that has the point with the minimum cost among them. Then we know that the isocurve of minimum cost will be within the areas on either side of this arc. So, we search the one or two slices of the surface that is/are adjacent to that arc. For example, if OE has the point with minimum cost we search the area BOC, and if OB has the point with minimum cost we search the area BOE.

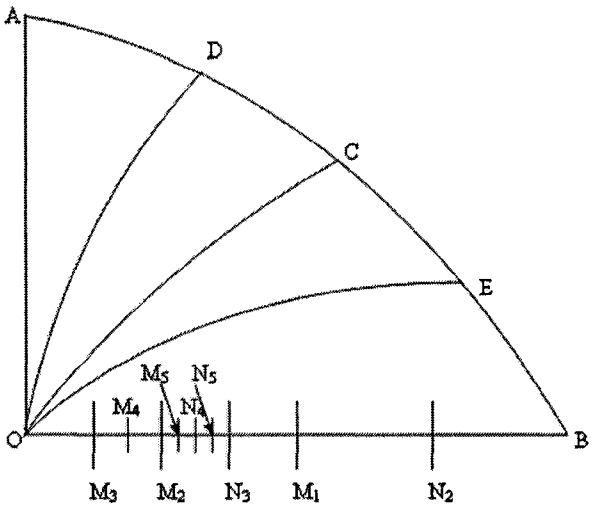

Figure 3.8: Binary slice and search of an octant AOB. The algorithm slices the domain using arcs and searches each arc using binary search method for 2D.

For searching BOE, we follow the method of searching AOB. For searching BOC, we search along the arcs that divide the slices COE and EOB using our binary search method for 2D up to the desired level of refinement. Then we select the arc that contains the point with minimum cost and apply the search again to the two refined slices adjacent to that arc. We continue this process until the thickness of the slice reaches to the desired level of refinement.

For refining the search space to $(1/2^5 \times 1/2^5)^{th}$ of the octant we need to use a maximum of $2 \times 5 + 1 = 11$ arcs and a maximum of $2 \times 5 + 1 = 11$ points on each arc. O is common to all the arcs. So, we need a maximum of $11 \times 11 - 10 = 111$ points to search an octant.

For the whole isosurface of motion constraints there will be more than one point of contact with local minima or maxima. Similarly to 2D, if an octant has more than one minimum or have both minimum and maximum we search the area that contain the absolute minimum for the octant. If an octant or its part has one local minimum the above

method will find it. If there is a local maximum, the local minima will be determined at an edge of the octant. We can apply the above search method separately to all the eight octants and the minimum cost point among all the eight resulting minimum cost points of the eight octants will be the solution point. We need to evaluate 8 x 111 = 888 points to search eight octants each with the refinement of $(1/2^5 \times 1/2^5)^{th}$ of the octant. This reduces the whole isosurface to the refinement of $(1/2^7 \times 1/2^7)$, or less than (1% x 1%) of its total area. There is one common edge for each pair of octants in each half surface for which the total number of points to be evaluated reduces by 2 x 4 x 2 x 5 = 80. Also, there is a common dividing curve which divides the surface into two. For this the total reduces by 4 x 5 = 20. Each of the the two halves has an apex point which is evaluated for each of the octants in its corresponding half. For these two points the total reduces by 6. Thus, the total number of distinct points to be evaluated is 782.

Similarly to 2D, we can use the properties of the isosurfaces to eliminate the octants that are known not to contain the absolute minimum for the whole isosurface of motion constraints, and find the resulting one to four octants of which one will contain the absolute minimum. If one octant or its part is selected we search that area using the binary slice and search method. We know that it will evaluate a maximum of 111 points.

If more than one octant is selected we can apply the binary slice and search method to each of them separately to find the absolute minimum for each of them. The minimum among them will be the solution point. For the maximum of four octants it will search a maximum of 401 points. To improve the performance we can identify the octant or part of an octant that contains the absolute minimum for the problem in a manner similar to 2D. For that we check the common edges of the octants to determine whether

the cost is decreasing towards the interior of an octant. If it is decreasing towards an octant that octant will contain the absolute minimum for the problem. Otherwise, the absolute minimum point is on the common edges or at the point of intersection of the isosurface of motion constraints with the axes of the isosurface of visual constraints. For the worst case of four octants we check a pair of opposite edges of the isosurface of motion constraints and their four adjacent arcs, at the level of refinement of the problem, with one on each side of them. We search these edges and arcs separately at the level of refinement of the problem using the binary search method for searching an arc. If the cost is decreasing towards the interior of an octant that octant will constain the absolute minimum for the problem. We search that octant using the binary slice and search method for searching a surface. This will evaluate 100 more points in addition to the 61 points that are evaluated for searching the two edges and their four adjacent arcs. Thus a total of 161 points are evaluated. But if the cost is not decreasing towards the interior of any octant we do not need to search any octant. The solution point will be on an edge or at a point of intersection of the isosurface of motion constraints with the axes of the isosurface of visual constraints.

Similarly to 2D, we use ordered set of numbers to represent the minimum cost slicing arc and the minimum point on the arc so that if the minimum cost solution is invalid for some reason we can backtrack and search. We need an ordered set to represent the hierarchy of slicing. Each of its elements contains the serial number of the arc whose minimum cost point is the lowest amongst the arcs in the corresponding pass. It organizes the slices in cost hierarchy that is similar to the hierarchical organization of subintervals

of an arc for 1D searching based on cost. For each slicing arc we need an ordered set of numbers that is the same as that for searching an arc for 1D searching.

3.8.1 Backtracking Search

We can improve the performance of the brute force search of an octant by using backtracking. Similar to the brute force search, in each iteration we divide the slice(s) adjacent to the minimum point arc in the previous iteration, and search a portion of the dividing arc(s) that contains the minimum point on that/those arc(s) with domain size of one half of the domain size on the dividing arc(s) in the previous iteration. So, we need to find either three points on this/these dividing arc(s) with the minimum cost in the middle point, or two points with the minimum cost at the end of the arc(s). We start the search for the domain on the dividing arc(s) by evaluating points near the minimum points in the previous iteration.

More specifically, to evaluate the first point on a dividing arc, if the average of the minimum point positions of its two adjacent arcs in the previous iteration is at one of the 5 (or 3) partitioning points that divide the domain of the minimum point arc into 4 (or 2), we first evaluate a point at the same level on the dividing arc(s) in the current iteration, then evaluate the point on this arc that is on the other side of the vertex of the quadrant and whose distance from the first point is one half of the domain size in the current iteration. If the average is not at the partitioning points, we round the fraction toward the partitioning point that is near the minimum point between the two adjacent arcs. Then, depending upon the cost at the vertex of the quadrant and these two points, and the length of the interval containing these three points, we evaluate a point on one side of these two

points at a distance of one half of the domain size from the nearest one. If an interval containing the minimum cost for the arc is found we search that interval using 1D binary search method up to the desired level of refinement. Otherwise, we evaluate a point next to and at a distance of half of the domain size from the previous point. We continue the process until an interval is found that contains the minimum point for the arc, or we reach an end of the arc. For the latter case, the interval at the end contains the minimum. Then we search the resulting interval using 1D binary search algorithm up to the desired level of refinement.

To show the backtracking, consider an octant AOB (Figure 3.9). Similar to the brute force slice and search method, first we search the arcs OA, OB and the dividing arc OC using 1D binary search method up to the desired level of refinement and find the minimum points on them. Let the minimum points be P, Q and R respectively. If the minimum among them is at P or Q, we need to search only one slice AOC or COB respectively, otherwise we need to search both the slices. Let the minimum be at R. Then we search the dividing arcs OD and OE using backtracking binary search. The domain size will be one half of the size of the arc in the previous pass. For this we first evaluate the points on OD

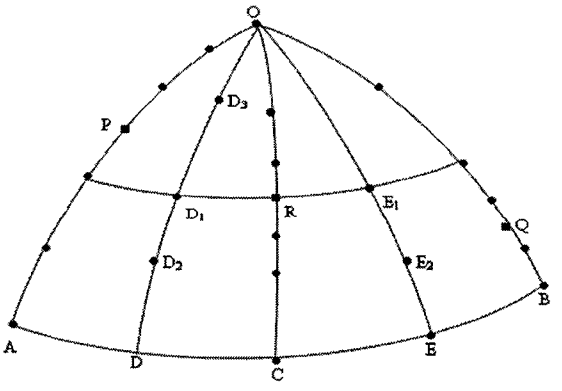

Figure 3.9: Backtracking search.

and OE that are near R. Here R is at the middle partitioning point of OC. We evaluate the mid-points D_1 and E_1. Then we evaluate D_2 and E_2 that are farther from O.

On OD, if D_1 is the minimum among O, D_1 and D_2, we evaluate the point D_3 between O and D_1. Then OD_3D_1 contains the minimum, so we search OD_3D_1 up to the refinement of the problem using 2D binary search. If D_2 is the minimum among O, D_1 and D_2 we evaluate D. Then D_1D_2D contains the minimum, so we search it using up to the refinement of the problem using 2D binary search. Similarly, on OE we select and search an interval of half of its size.

In this backtracking method, starting from OD and OE, the search domain is reduced to one half in each iteration. Since the isosurfaces are convex and O is not the minimum point, in the worst case the algorithm may start with the minimum points near O or AB, and backtrack to the other end. The backtracking may proceed gradually in many iterations or abruptly in one iteration. For abrupt backtracking we double the step of backtracking at each point. Suppose we want to search up to the granularity of $(1/2^5 \times 1/2^5)^{th}$ of the octant. Searching OA, OB and OC requires $3 * (1 + 2 * 5) - 2 = 31$ points to be evaluated. In each iteration the backtracking method searches one half of the domain size in the previous iteration and backtracks on the average of one point. So, for gradual backtracking at most $2 * [2 * (4 + 1) + 2 * (3 + 1) + 1 + 2 * (2 + 1) + 1 + 2 * (1 + 1) + 1] = 62$ points will be evaluated. For each octant, at most 93 points are evaluated. On the other hand, the brute force binary slice and search algorithm takes 111 points to search an octant. Thus, this is not a big improvement to the brute force method.

3.9 Example

We have implemented our framework in the 2D and 3D tracking shot of a single target moving in an environment without any occluder or other environment elements. For 2D we have used four constraints such as distance and orientation of the camera with respect to the target, and frame coherence distance and rotation. For 3D we have included two additional constraints such as vertical angle of the camera with respect to the target and the frame coherence vertical rotation. Each has a desired value and a range of acceptable values. The framework uses the range of values to find the weights for the respective constraints automatically. The resulting camera position and its motion are found to be as expected.

3.10 Discussion

In this chapter we have described our approach to automatic camera control using a simple tracking shot of a single target in 2D and 3D without any occluder. We use a weighted constraint representation for the camera control problem. To keep it simple, we consider only four constraints for 2D such as frame coherence distance, frame coherence orientation, visual distance and visual orientation, and two additional constraints for 3D such as frame coherence vertical rotation and visual angle, and apply them purely reactively to enable it for a dynamic environment. Each of these constraints has an optimal value and a range of acceptable values. They give rise to the weights for them and identify the two systems of isocurves for 2D or isosurfaces for 3D. Then, finding the minimum cost solution reduces to finding the point of contact of two curves/isosurfaces from the two systems. But there is no known exact solution for this. So we use a binary

search technique along a curve for 2D and on a surface for 3D to find the minimum cost solution. The algorithm searches a maximum of 13 points to reduce the granularity of the domain to less than 1% for 2D. For 3D, it evaluates a maximum of 161 points for a brute force search to achieve the refinement of less than 1% x 1%. There is no appreciable improvement in efficiency in the backtracking search.

The optimal value and the range of allowable values of a constraint determine the weight for it. The relative priority value used in Bares et al. (2000a, 2000b) is not necessary among constraints within visual or motion constraints. Because increasing (or decreasing) the relative priority value from 1.0 is equivalent to decreasing (or increasing) the range of acceptable values for the constraint, provided the costs at the end points of the range of allowable values of each constraint are equally acceptable and we use the total of weighted constraint cost with constant weight for the cost function. However, the relative priority between the groups of visual and motion constraints can be used at a higher level to guide the motion of the camera. For this we do not need any specific value. There are only three cases, such as higher, lower or equal priority.

The seven parameters of the camera can be specified in the following way:

- First we find the best possible position for the camera satisfying frame coherence, distance, orientation, height and all the other constraints.
- Then we point the camera towards the centre of view.
- Finally, we adjust the focal length.

Chapter 4

Occlusion and Collision Avoidance

4.1 Introduction

There are two types of occlusion constraints:

- If the subject, object or the environment is required to be occluded wholly or partially for visual effects then the desired values and/or domains of the visual constraints such as distance, orientation and angle will be affected. So, this type of occlusion will have an effect on the determination of the weights of those visual constraints. In this thesis we shall not consider this type of occlusion.
- On the other hand, if the subject, object or environment are required to be unoccluded, then this will not have any effect on the quality of the shot and hence on the visual constraints such as distance, orientation and angle. The only effect this type of occlusion has is to make a potential position, or a part or the whole domain of potential positions of the camera either valid or invalid. This type of occlusion cannot be readily included as a constraint in the fitness function of the constraint satisfaction problem, because this constraint has no cost involved. This

constraint should have the effect of avoiding the occluded areas of the domain of camera positions. In this thesis we shall consider this type of occlusion.

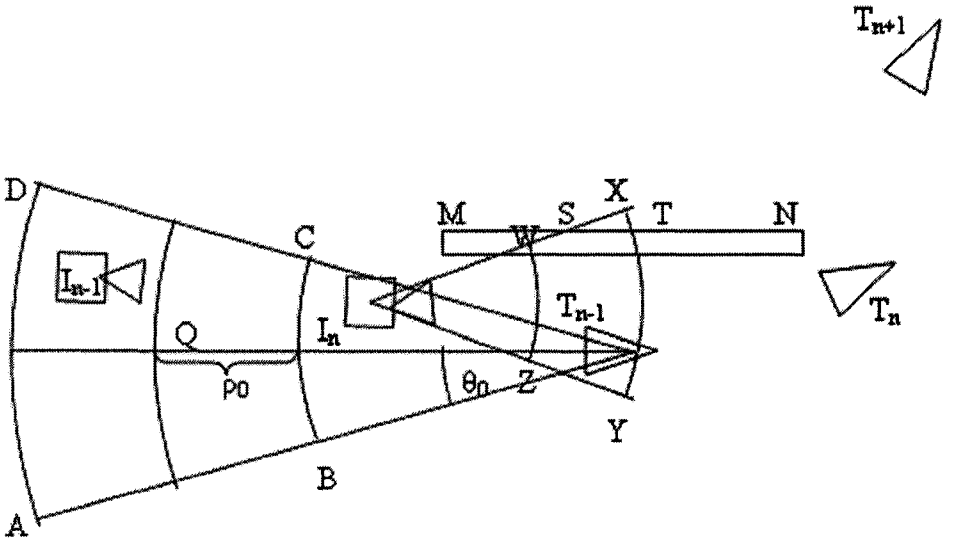

Figure 4.1: Unoccluded area SXT will not be a valid solution in the next frame due to collision with the object MN.

In Figure 4.1, let the n^{th} frame be the current frame. Let I_{n-1} and T_{n-1} be the position of the camera and the target in the previous frame. Let the target be at T_n in the current frame and assume that in the next frame (n+1) it will be at T_{n+1}. Let MN be an object that may cause occlusion. The object MN does not affect the visual quality of the shot or the motion of the camera in the current frame n. Suppose, if we do not consider the probable occlusion by MN in the future frame, the camera goes to position I_n in the current frame. If the camera can move to the area WXYZ in the next frame (n+1) according to the frame coherent motion, then from the figure we see that either the camera will collide with the object MN or the target will be occluded by that object in the $(n+1)^{th}$ frame. Although the area SXT of the region of frame coherent motion of the

camera is unoccluded, the camera cannot move there due to collision with the object MN. So, we need to consider occlusion avoidance and collision avoidance together.

To avoid occlusion and collision we can choose one of the following options:

- Cut to a new shot
- Continue with the same shot but move the camera to another point of view such as in front of the subject.
- If possible, continue with the same point of view and the same shot.

To cut to a new shot we need to know beforehand that the present shot cannot be continued longer, and that we have to wait for the appropriate action or situation in the scene when we can cut to a new shot. During this waiting period the camera configuration for the new shot can be computed. So, we need a prediction about the future state of the scene and the camera. Otherwise, the cut may not be smooth, we may miss the best shot that we can shoot next if we cut immediately to a new shot before that, and we may not have enough time to compute the camera configuration for the next shot.

If we can guide the camera to avoid the occlusion and the collision, and we do not have the appropriate state of the scene to cut to a new shot or we do not need to cut to a new shot, then we can continue with the present shot.

To move the camera to a new point of view or to continue with the present point of view in the same shot we need to avoid the occlusion or collision of the camera with the environment or other objects or subjects.

In this thesis we discuss the occlusion or collision avoidance while continuing with the same point of view. Changing the point of view will involve determining the new point of view. Once a new point of view is selected, moving the camera towards that location is the same as moving the camera for the same point of view, except that the point of view for the current frame will be located at a new position relative to the subject of the shot.

It is desirable to avoid occlusion even for a single frame. In some situations, a brief occlusion for a single or a couple of frames, viz., occlusion by a fast moving object for a couple of frames, may be acceptable (Hawkins, 2000). But collision cannot be allowed even for a single frame. So, the camera needs to have knowledge about the future situation of the scene so that it can avoid occlusion and collision or it can cut to a new shot. Motion characteristics of the subject, object and the camera can be used to predict their future positions.

But, for a completely dynamic environment such as computer games there will always be some inaccuracy in the prediction. Since we use prediction at each frame, any deviation from the predicted positions due to the dynamic nature of the problem will be corrected by the predictions in the next frames.

As the number of frames for prediction increases, the inaccuracy of the predicted information increases. But since we need to have future information about occlusion and collision anyway, we shall use prediction for a couple of frames. We can use past velocity, angular velocity, acceleration and angular acceleration of the subject, camera and object to predict about their future position. In this thesis we shall not be concerned

with developing an efficient method for prediction. We shall use only their linear and angular velocity to roughly predict their future position and orientation and discuss how to avoid probable occlusion and/or collision by assuming that they will be around those predicted positions.

4.2 Effect of Occlusion and Collision Avoidance on Constraints

Occlusion and collision are related to camera motion. Occlusion and collision do not affect the visual quality of a shot. They simply make some areas of the world invalid. But the cost of visual constraints at the unoccluded and collision free areas will remain unchanged. So, we cannot change the weightage for the visual constraints. Since the weight of a constraint is inversely proportional to the range of acceptable values for that constraint, it follows that the domain of visual constraints cannot be changed, otherwise the cost of the visual constraints will be modified inappropriately. We cannot restrict the domain of visual constraints to guide the camera to move more vigorously towards the unoccluded or collision free areas. So, they will restrict the domain of camera motion. This will in turn increase the weights of the corresponding motion constraints. Thus, the camera will be moved vigorously away from the possible occlusion and collision area.

If different parts of the motion domain have the problem of occlusion and/or collision and we want to move the camera through that area (e.g., moving the camera through the bushes or the leaves of a tree), we cannot restrict the domain to guide the camera towards an occlusion and/or collision free area. For such cases we need to search the whole motion domain. If there is a collision problem at different parts of the motion domain, the minimum point may have an occlusion and/or collision problem. The next

best solution may be in the interior of the domain of motion constraints. We also need to search the interior area of the domain of camera motion. Our binary search algorithm can not search the interior efficiently. In this thesis we shall not consider the avoidance of such type of collision. We shall discuss the searching method for the other cases when there is occlusion at different parts of the motion domain, but no collision problem therein.

4.3 Occlusion and Collision Avoidance in the Current Frame

First we discuss the avoidance of occlusion and collision in the current frame only. Here we assume that either there will be no occlusion or collision problem in the predicted future frames or that we do not need to consider the occlusion and collision problem in the future frames (this is possible for such cases as when the current frame will be the last frame of this shot). For this case we do not need to guide the camera to move towards the area that is promising for the future frames. We just need to avoid the occlusion and collision areas from the domain of motion constraints.

First we consider collision avoidance (Figures 4.2a and 4.2b). In these figures, I is the position of the camera in the past frame, Q is the desired position of the camera according to the motion constraints, MN is the probable colliding object and ABCD is the domain for motion constraints. ρ_0, θ_0 and φ_0 are the range of acceptable values for motion constraints ρ, θ and φ respectively. We use hardware rendering to project the region of camera motion IBC on the farthest arc (for 2D) or surface (for 3D) CB, with the centre of projection being at the position of the camera I. We can use, say, 32 x 32 buffer in the rendering.

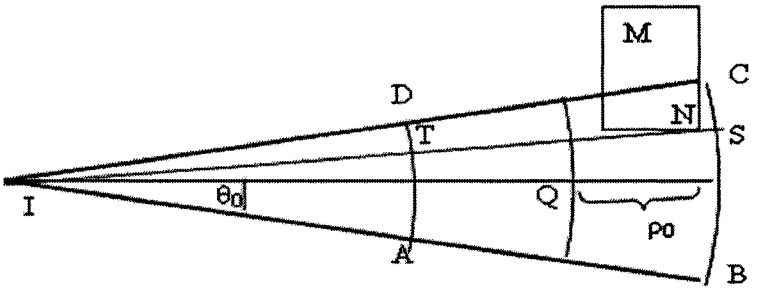

Figure 4.2a: Collision avoidance without prediction: Desired position of camera is not affected by collision problem.

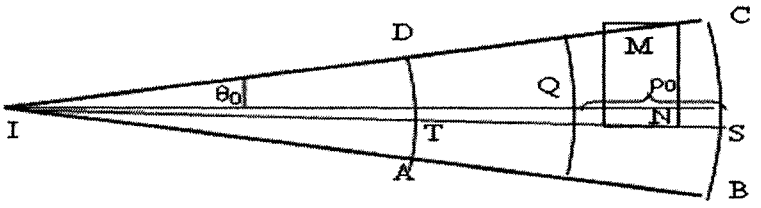

Figure 4.2b: Collision avoidance without prediction: Desired position of camera is affected by collision problem.

In Figure 4.2a and 4.2b, CS is the projection of the object MN on CB. In Figure 4.2a, positions along the desired orientation have no collision. The desired orientation and the range of orientation for the lower half domain for the motion constraints will remain unchanged, the range of orientation for the upper portion of the domain will be reduced to the angle SIQ. In Figure 4.2b, position along the desired orientation also has collision. We change desired orientation to the direction of IS, and change the range of values for orientation for the top and bottom portion of the domain to 0 and the angle SIB respectively.

If different areas of BATS (or different areas of ABCD if the object MN does not cause any collision problem) have a collision problem, those areas will be identified in

the projection. We mentioned before in Section 4.2 that this kind of collision problem will not reduce the domain size. But occlusion can reduce the size of the motion domain. So, we can consider this kind of collision avoidance together with a similar kind of occlusion avoidance. As mentioned in Section 4.2, we shall not consider this kind of collision.

After the application of the condition of collision avoidance, the restricted motion domain is valid for movement of the camera to any point of the restricted domain of motion that is not identified as colliding. Now we apply the occlusion avoidance condition to further restrict that domain.

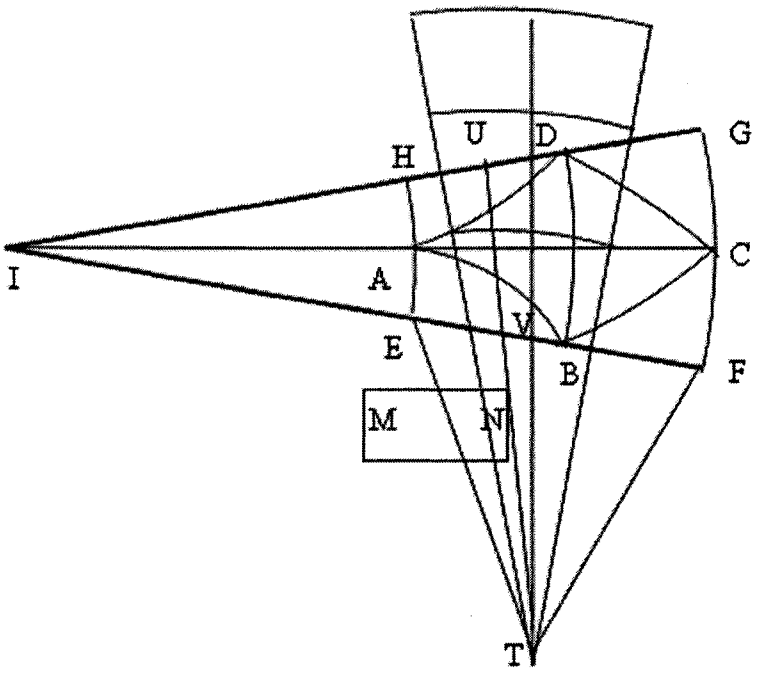

Figure 4.3: Occlusion avoidance without prediction: Arc UAV of the isocurve of motion constraints is occluded by the object MN. It is removed from the search domain.

Let I be the position of the camera in the last frame and T be the position of the target in the current frame (Figure 4.3). We assume that we have already validated the collision avoidance condition which has restricted the motion domain to EFGH. First we

consider the case that no area of EFGH has a collision problem. So, there is no object within this domain to cause any occlusion to the target. We need to check for the occluder in the area ETF. We project the area between the isocurve ABCD and T onto the isocurve with T as the centre of projection. Let TN intersects with the isocurve ABCD at V and U. The area VAU of the isocurve is occluded and the area VBCDU of the isocurve is unoccluded. We search this unoccluded area of the isocurve and the part of TN cut by it, i.e., the curve VBCDUV to find the minimum cost solution.

If some areas of VBCDU (or some areas of the whole isocurve ABCD - if the object MN did not cast any occlusion) are occluded we use backtracking guided by the ordered set of numbers for hierarchical representation of the curve/surface to find the unoccluded best cost solution. For this search we use binary search for 2D problems or binary slicing and searching for 3D problems and generate the ordered set of numbers. Then we check the minimum cost solution for occlusion by checking the buffer array for occlusion. If it is not occluded then it is the solution. Otherwise, we need to backtrack and search. For 2D problems we check the buffer array for unoccluded points on both sides of the minimum point and get the location of those two points. We evaluate the visual cost of visual constraints for them. The one with the minimum cost will be the solution. For 3D we use the ordered set of numbers to find the area of next best cost solutions and check that area in the buffer array for occlusion. If there is an unoccluded area in that region, we select this area. If the next best cost region is found to be occluded, we use the ordered set of numbers to find the area of next best cost solutions. We repeat this procedure until we find a region of best cost points that has unoccluded points. We use

the binary slice and search to search that area up to the desired refinement of the search space. Then, the buffer array is checked for occlusion and the process is repeated.

If during the collision avoidance procedure some areas of VBCDU (or some areas of the whole isocurve ABCD – if the object MN did not cast any occlusion) have been found to be invalid due to collision problem, irrespective of whether some of its area is now found to be invalid due to occlusion problem, the least cost solution on the surface of the isocurve of motion constraints may become invalid. Some points interior to the isocurve may be the next best cost and valid solution. So, we need to search the interior of the motion isocurve. As mentioned before in Section 4.2, our binary search algorithm can not search that area efficiently, and so we shall not consider this type of collision in this thesis.

4.4 Variable Weight

Now let us consider another example of occlusion and collision avoidance (Figure 4.4). Let KLMN be a wall with NM on the ground and KL being the top of it. Let KN be vertical and LM be inclined. Here the domain of θ is changing vertically and that of φ is changing sideway θ_0 is decreasing downward and φ_0 is decreasing leftward. They may become 0 at some point. Further down the point where $\theta_0 = 0$ the desired values for θ will be the values of θ for the points along the line LM. Similar will be the case for the desired values of φ for further left of the point where $\varphi_0 = 0$. This gives rise to variable θ_0 and φ_0. In other cases ρ_0 may be variable. Still the isosurfaces are convex and isosurfaces of higher cost will contain those of lower cost. So, we can use our binary slice and search algorithm on the isocurve of motion constraints to find the solution.

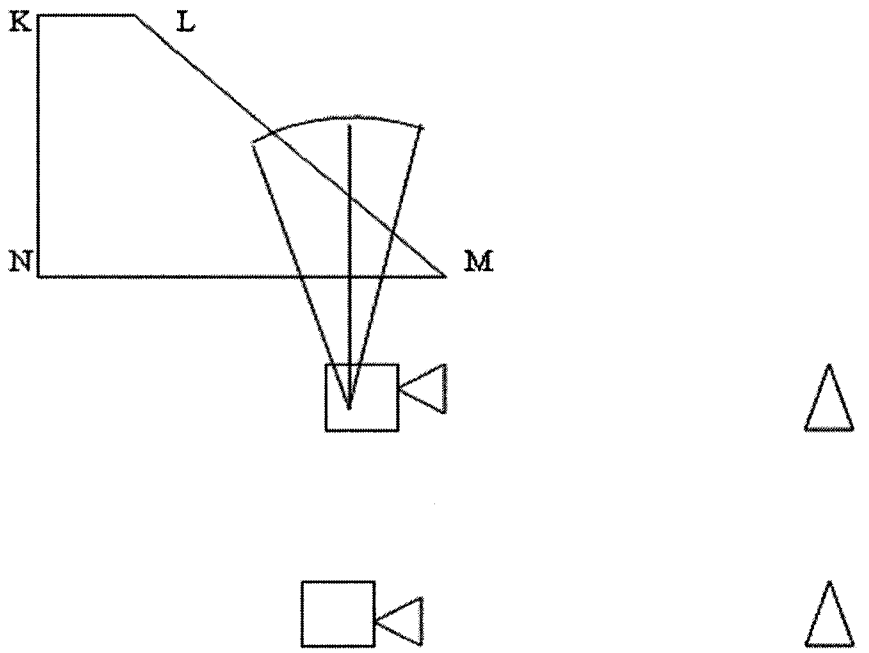

Figure 4.4: Variable weight.

4.5 Using Prediction to Avoid Occlusion and Collision

Now we discuss the avoidance of probable occlusion or collision in the future frames. In Figures 4.5a, 4.5b, 4.6a and 4.6b, the target is at T_{n-1}, T_n and T_{n+1} in the past frame (n-1), current frame n and next frame (n+1). Let the camera be at I_{n-1} in the previous frame. If we do not use prediction, suppose the camera moves to I_n in the current frame. Then in the next frame (n+1), the camera will collide with the object MN in Figures 4.5b, 4.6a and 4.6b. For Figure 4.5a, either the camera will collide with the object MN or the target will be occluded by the object MN in the next frame (n+1). We shall use hardware rendering to check for occlusion and collision in each frame.

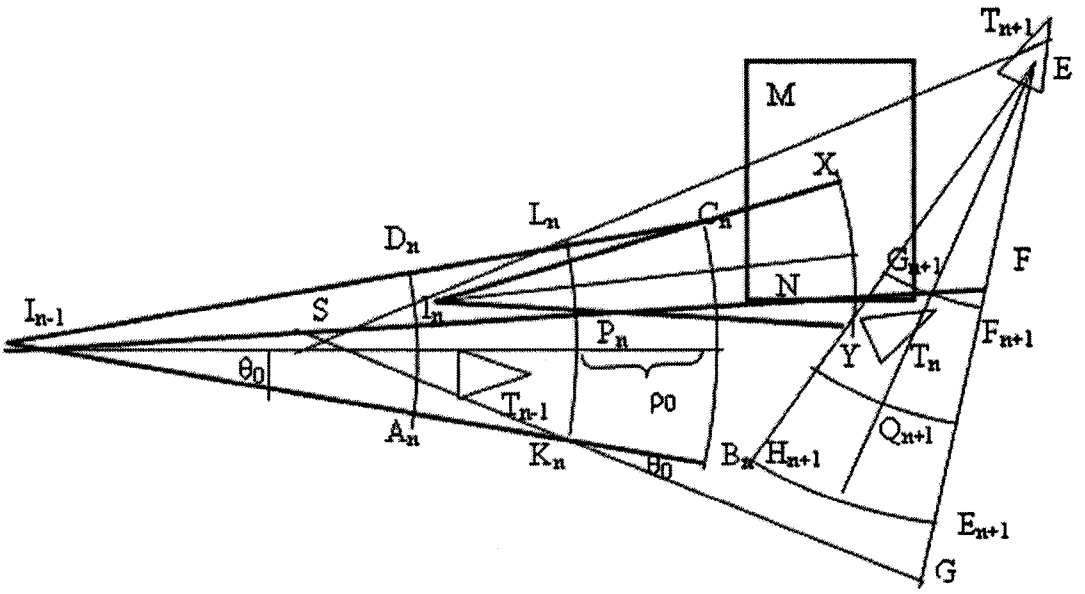

Figure 4.5a: Desired position P_n according to motion constraints is a valid solution.

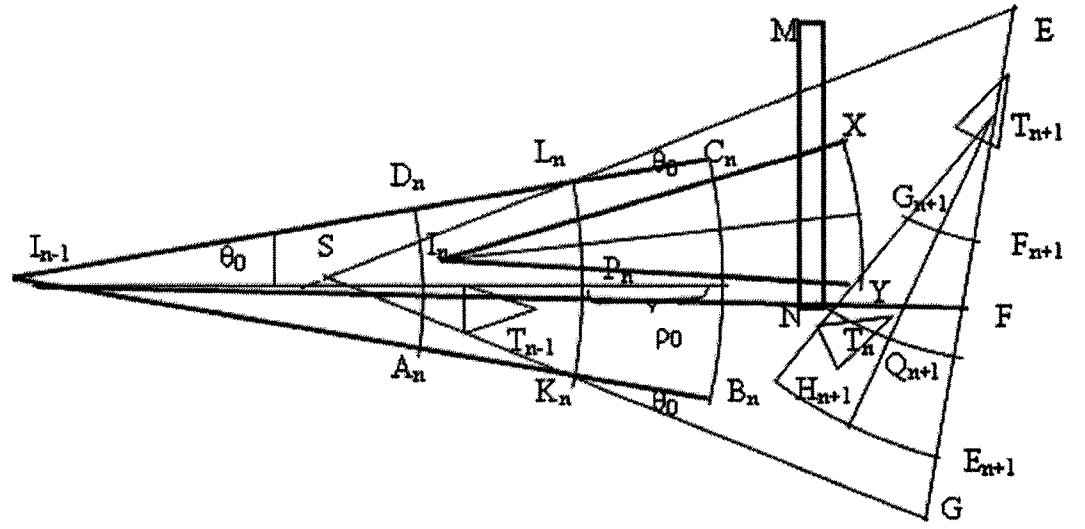

Figure 4.5b: Desired position P_n according to motion constraints is not a valid solution due to collision problem in the next frame.

Figure 4.5: Avoiding collision in the next frame for an over the shoulder tracking shot.

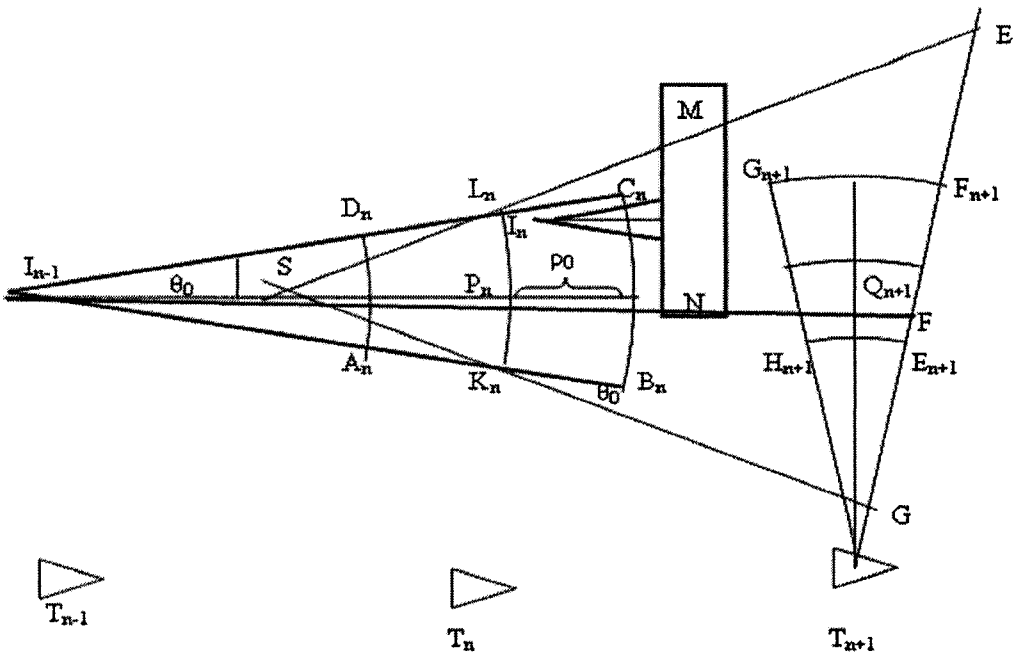

Figure 4.6a: Desired position P_n according to motion constraints is not a valid solution due to collision problem in the next frame.

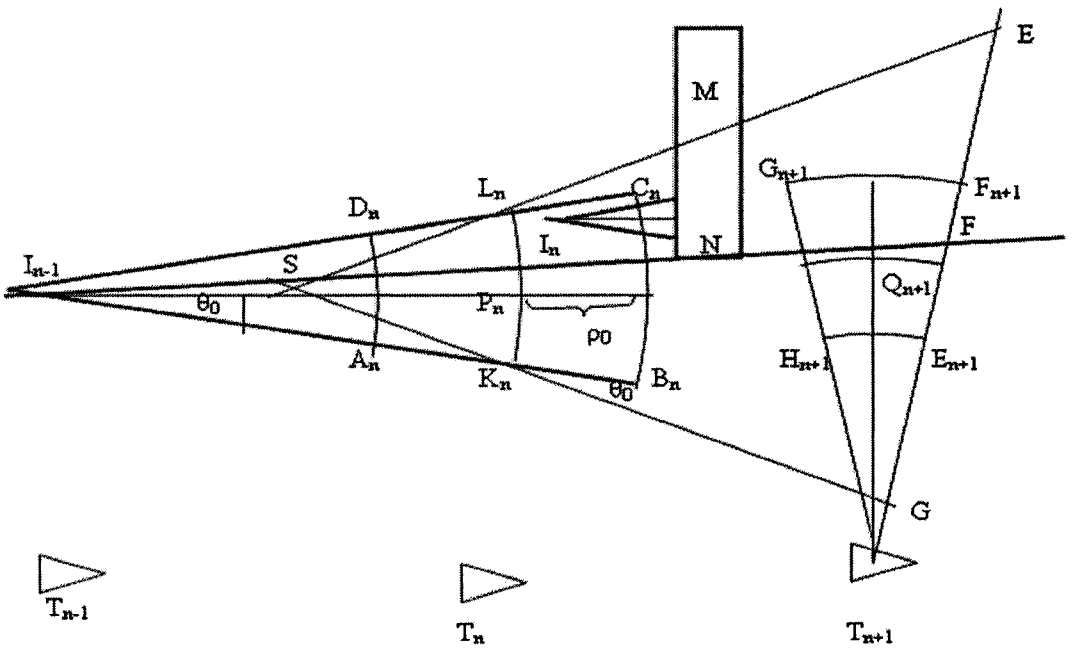

Figure 4.6b: Desired position P_n according to motion constraints is a valid solution.

Figure 4.6: Avoiding collision in the next frame for a profile tracking shot.

First we check for occlusion and collision in the current frame. To check for collision we use hardware rendering to project the region $C_nB_nI_{n-1}$ on B_nC_n using I_{n-1} as the centre of projection. We see from Figures 4.5a, 4.5b, 4.6a and 4.6b that there is no collision in the current frame. Let L_nK_n be the arc for the desired value of distance. Then, the maximum angular stretch for the camera position in the next fame (n+1) will be within the region ESG. Using hardware rendering we project the region ESG on the line $T_{n+1}F_{n+1}$ with S as the centre of projection. Let the portion EF of this projection be occluded. If the desired position P_n for motion constraints collides with MN (Figures 4.5b and 4.6a), then our desired orientation for the angular movement will be along $I_{n-1}NF$, and the range of acceptable values of θ for the upper region would be 0 and for the lower region would be (θ_0 - angle ($P_nI_{n-1}N$)). If P_n does not collide with MN (Figures 4.5a and 4.6b) we do not need to modify the desired value for orientation and its range of values for the bottom part. We can avoid the collision by changing the range of acceptable values of θ for the upper part to the angle $FI_{n-1}P_n$.

Here we have used prediction about only one future position of the target at T_{n+1} and occluder at MN. We do not use prediction about the future position of the camera. Similarly, using prediction about a couple of future positions of the target, we can increase the ability of the camera to avoid collisions which cannot be avoided by using prediction about only one frame. If it is seen by prediction that collision cannot be avoided in the future, then we will have time to change the point of view or cut to a new shot.

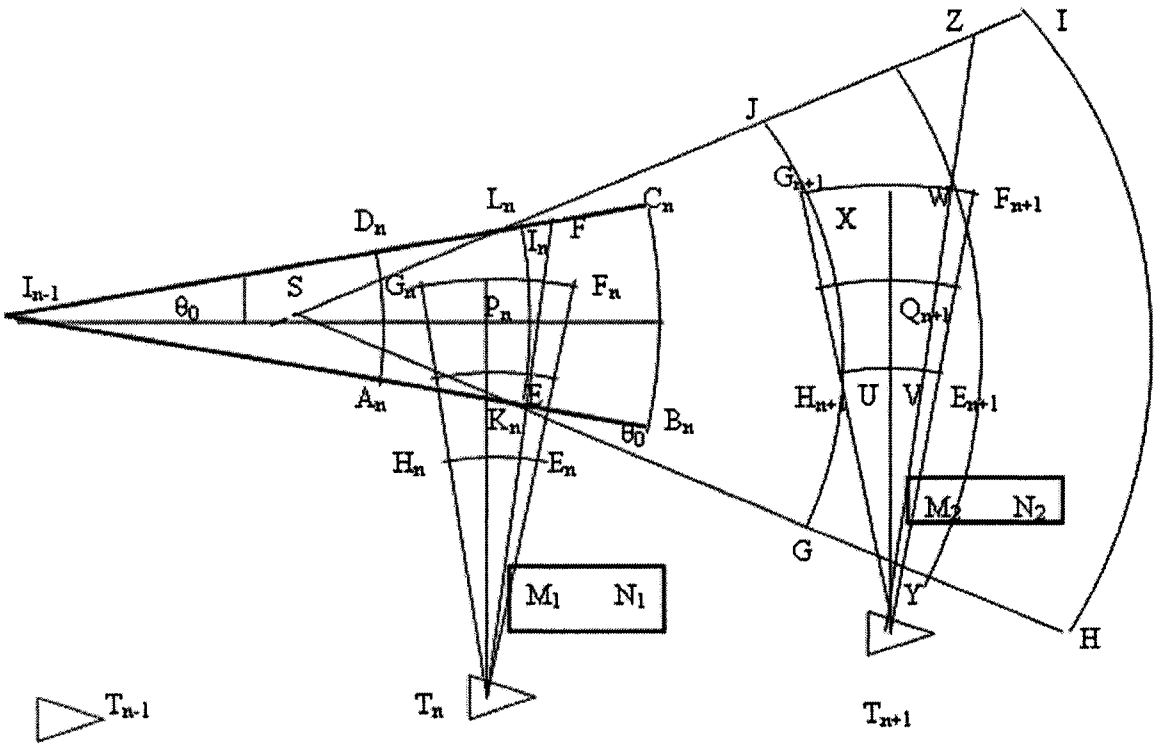

Figure 4.7: Occlusion avoidance in the next frame using prediction.

Now we discuss occlusion avoidance using prediction. Here we shall consider prediction for one frame only. We predict about the position of the target and the occluding objects in the next frame based on their motion in the current frame. Let T_{n-1} and I_{n-1} be the position of the target and the camera in the previous frame (Figure 4.7). Let T_n be the position of the target in the current frame. Let T_{n+1} be the predicted position of the target in the next frame (n+1) based on its motion and $E_{n+1}F_{n+1}G_{n+1}H_{n+1}$ be the domain for visual constraints for the next frame. Let M_1 N_1 be the position of an occluder in the current frame. Let M_2 N_2 be the predicted position of another occluder based on its motion in the previous frame.

Let $A_nB_nC_nD_n$ be the domain of camera motion in the current frame after the restriction imposed by the collision problem. Occlusion by M_1 N_1 restricts the domain to

A_nEFD_n. With this domain in the current frame, the camera can move to the area GHIJ in the next frame. Occlusion by M_2 N_2 in the next frame restricts this range to GYZJ which intersects with the visual domain $E_{n+1}F_{n+1}G_{n+1}H_{n+1}$ in the next frame at UVWX. We evaluate the domain of camera positions in the current frame in order to move to the area UVWX in the next frame according to the motion of the camera. Now we adjust the desired position of the camera for the motion constraints. This will identify the isocurve for the motion constraints. We apply the search on this curve.

If the common area UVWX does not have any valid position due to a scattered occlusion or collision problem, the next frame may not have a solution camera position in the current point of view. We then prepare to change the point of view or cut to a new shot.

Chapter 5

Conclusion

5.1 Summary

Due to its generality the constraint based approach is used to solve the autonomous camera control problem in an interactive digital environment. Conflicting requirements make the problem over-constrained. Difference of priorities for the constraints warrants the representation of the cost function as a weighted sum. Initially a simple tracking shot of a single target in an environment with no occluder is considered.

The total weighted cost of the problem is decomposed to decouple the frame coherence cost. It is shown that the weight for a constraint is determined by the acceptable range of values for it. This relieves the user of specifying the weights. The search space is represented as two systems of isocurves/isosurfaces with constant weights corresponding to the two types of constraints such as visual and motion constraints.

The minimum cost solution is on the locus of the point of contact of the two systems of curves/surfaces. If the visual and frame coherence constraints are considered as equal in importance, all the points on this locus have equal cost, so any point on this locus is the minimum cost solution. If any of these two types of constraints has a higher

priority compared to the other, the most optimal position with respect to that particular type of constraint will have the minimum cost solution. Since it is always desirable to accelerate or decelerate the camera to reach the optimal position with respect to the visual constraints, the visual constraints must be given higher priority than the motion constraints. There is no need to specify the relative priority value. It only makes the solution point lie on the point of contact of the outer isocurve/isosurface for the motion constraints with the corresponding isocurve/isosurface for the visual constraints. This has the advantage of providing the autonomous camera a higher level control on the linear and angular speed of the camera by increasing or decreasing the range of acceptable values for their respective constraints.

The point of contact of the two curves/surfaces from the two families of isocurves/isosurfaces is the exact solution for the optimal camera position. Since camera motion constraints has the lower priority within a certain region of space around its desired position, the solution camera position will be at the point of contact of the isocurve/isosurface of motion constraints which encloses that area with an isocurve/isosurface of the visual constraints. This region contributes to the acceleration or deceleration of the camera which can be controlled from a higher level to move the camera in an informed way to the desired location.

The method of finding the exact solution for the point of contact is not known. So, we use a binary search technique to search on the isocurve/isosurface of motion constraints. For 2D it evaluates 13 points to reach the granularity of less than 1% of the search domain. For 3D it searches a maximum of 161 points. The search organizes the search space in a hierarchy of cost. When the minimum cost solution is not a valid

solution and we need to find the nearest best cost solution, we can use the hierarchical information for cost to direct the search to the next best solution.

We use prediction to determine the occlusion and collision. Hardware rendering is used to project the area of interest for occlusion and collision information. Except for the case when different parts of the motion domain have the problem of occlusion and collision, the occlusion and collision problems are tackled by removing the invalid collision and occluded area from the domain of camera motion. They affect the isocurve/isosurface of motion constraints and the search area respectively.

5.2 Future Work

Future research can be undertaken in the following directions:

- Our binary search method cannot search the interior of the search domain efficiently. An efficient method can be formulated to do that.
- Until now, constraints for camera control have not been methodically studied. They can be investigated.
- Variable constraint weights and irregular bounding surfaces for the domains of visual and motion constraints can be investigated.
- More efficient and reliable techniques to predict about two or more future frames can be studied.
- Determination of the maximum limits of the radial and angular acceleration and deceleration of the camera in relation to its radial and angular speed and visual requirements can be investigated.

References

1. Arijon, D. (1991) *Grammar of the Film Language.* Silman-James Press, September, (Originally: Hastings House Publishers, 1976).
2. Bares, W. H., Gregoire, J. P. and Lester, C. J. (1998) Realtime Constraint-Based Cinematography for Complex Interactive 3D Worlds. In *Proceedings of Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference (AAAI-98/IAAI-98)*, Madison, WI, USA, July 1101-1106.
3. Bares, W. H. and Lester, J. C. (1999a) Intelligent Multi-Shot Visualization Interfaces for Dynamic 3D Worlds. In *IUI '99: Proceedings of the 4th International conference on Intelligent User Interfaces*, New York, USA, ACM Press, 119-126.
4. Bares, W. H. and Lester, J. C. (1999b) Intelligent Multi-Shot 3D Visualization Interfaces. *Knowledge-Based Systems*, 12(8), 403–412.
5. Bares, W., McDermott, S., Boudreaux, C. and Thainimit, S. (2000) Virtual 3D Camera Composition from Frame Constraints. *MULTIMEDIA '00: Proceedings of the Eighth ACM International Conference on Multimedia,* ACM Press , 177-186.
6. Bares, W. H., Thainimit, S. and McDermott, S. (2000) A Model for Constraint-Based Camera Planning. *AAAI2000 Spring Symposium Series on Smart Graphics,* Stanford, California, USA, March, 84-91.

7. Barwood, H. (2000) Cutting to the Chase: Cinematic Construction for Gamers. *Game Developers Conference*, San Jose, CA, USA. http://www.gamedeveloper.com/features/20000518/barwood_pfv.htm.
8. Blinn, J. (1988) Where am I? What am I Looking at? *IEEE Computer Graphics and Applications*, July, Volume 8, 76–81.
9. Bourne, O. and Sattar, A. (2004a) Applying Constraint Satisfaction Techniques to 3D Camera Control. In M. Wallace, ed., *Principles and Practice of Constraint Programming* (CP2004), LNCS 3258, Toronto, Canada. Springer-Verlag, Berlin, Heidelberg, 811.
10. Bourne, O. and Sattar, A. (2004b) Applying Constraint Satisfaction Techniques to 3D Camera Control. In G. L. Webb and X. Yu, eds., *17th Australian Joint Conference on Artificial Intelligence,* Cairns, Australia, December. Springer, 658-669.
11. Bourne, O. and Sattar, A. (2005a) Applying Constraint Weighting to Autonomous Camera Control. *Artificial Intelligence and Interactive Digital Entertainment*, Marina Del Ray, CA, USA, June. AAAI Press, 3-8.
12. Bourne, O. and Sattar, A. (2005b) Evolving Behaviours for a Real-Time Autonomous Camera. *Proceedings of the Second Australasian Conference on Interactive Entertainment,* Sydney, Australia, ISBN 0-9751533-2-3/05/11, 27-33.
13. Bourne, O. and Sattar, A. (2006) Autonomous Camera Control with Constraint Satisfaction Methods. In S. Robin, ed., *AI Game Programming Wisdom 3,* Charles River Media, 173-187.
14. Bourne, O. (2006) Constraint-Based Intelligent Camera Control for Interactive Digital Entertainment. *PhD Thesis*, Institute of Integrated and Intelligent Systems, Griffith University, Queensland, Australia, March.

15. Christie, M. and Languenou, E. (2003) A Constraint-Based Approach to Camera Path Planning. *Smart Graphics, Third International Symposium, SG 2003,* eds. A. Butz, A. Kruger, and P. Olivier. Volume 2733, Lecture Notes in Computer Science, Springer, 172-181.

16. Christie, M., Languenou, E. and Granvilliers, L. (2002) Modeling Camera Control with Constrained Hypertubes. In Pascal Van Hentenryck, ed., *Principles and Practice of Constraint Programming (CP2002)*, Ithaca, New York, USA, Springer, 618-632.

17. Christie, M., Machap, R., Normand, J-M., Olivier, P. and Pickering, J. (2005) Virtual Camera Planning: A Survey. In Andreas Butz, Brian Fisher, Antonio Krüger, Patrick Olivier, eds., *Smart Graphics: 5th International Symposium, SG 2005*, Frauenwörth Cloister, Germany, August. Lecture Notes in Computer Science, Springer-Verlag, Volume 3638/2005, 40. http://ifgi.uni-muenster.de/~kruegera/sg05/37.pdf

18. Christie, M. and Normand, J.-M. (2005) A Semantic Space Partitioning Approach to Virtual Camera Control. *Proceedings of the Annual Eurographics Conference*, Volume 24, 247–256.

19. Christie, M. and Olivier, P. (2006) Camera Control in Computer Graphics. In *Eurographics 2006,* eds. E. Gröller and L. Szirmay-Kalos (Guest Editors), Volume 25, Number 3, 1-25.

20. Christianson, D. B., Anderson, S. E., He, L., Salesin, D. H., Weld, D. S. and Cohen, M. F. (1996) Declarative Camera Control for Automatic Cinematography. *Proceedings of the American Association for Artificial Intelligence*, 148-155.

21. Drucker, S. M., Galyean, T. A. and Zeltzer, D. (1992) Cinema: A System for Procedural Camera Movements. *SI3D '92: Proceedings of the 1992 Symposium on Interactive 3D Graphics*, New York, NY, USA, ACM Press, 67–70.

22. Drucker, S. M. (1994) Intelligent Camera Control for Graphical Environments. *PhD Thesis*, Massachusetts Institute of Technology, June.

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

24. Drucker, S. M. and Zeltzer, D. (1995) CamDroid: A system for Implementing Intelligent Camera Control. *SIGGRAPH Symposium on Interactive 3D Graphics*, Monterey, CA, USA, April, 139-144.

25. Feiner, S. and Seligmann, D. D. (1992) Cutaways and Ghosting: Satisfying Visibility Constraints in Dynamic 3D Illustrations. *The Visual Computer,* August, 292-302.

26. Friedman, D. A. and Feldman, Y. A. (2004) Knowledge-Based Cinematography and its Applications. *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004*, Valencia, Spain, August, IOS Press, 256-262.

27. Gleicher, M. and Witkin, A. P. (1992) Through-the-Lens Camera Control. *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1992*, Chicago, IL, USA, 331-340.

28. Halper, N., Helbing, H. and Strothotte, T., (2001) A Camera Engine for Computer Games: Managing the TradeOff Between Constraint Satisfaction and Frame Coherence. *Proceedings of the Eurographics '2001 Conference*, eds. A. Chalmers and T.-M. Rhyne, Manchester, UK, September, 20(3), 174-183.

29. Halper, N. and Olivier, P. (2000) CAMPLAN: A Camera Planning Agent. *Smart Graphics 2000 AAAI Spring Symposium*, Stanford, CA, USA, March, 92-100.

30. Hawkins, B. (2005) *Real-Time Cinematography for Games*, Charles River Media.

31. He, L. W., Cohen, M. F. and Salesin, D. H. (1996) The Virtual Cinematographer: a Paradigm for Automatic Real-Time Camera Control and Directing. *Proceedings of 23rd AnnualConference on Computer Graphics (SIGGRAPH 96),* New Orleans, LA, USA, August, ACM Press, 217-224.

32. Jardillier, F. and Languenou, E. (1998) Screen-Space Constraints for Camera Movements: the Virtual Cameraman. In N. Ferreira and M. Göbel, eds., *Proceedings of EUROGRAPHICS'98*, Computer Graphics Forum, Volume 17, Blackwell Publishers, 108 Cowley Road, Oxford, OX4 IJF, UK, 175-186.

33. Katz, S. (1991) *Film Directing Shot by Shot: Visualizing from Concept to Screen*, Michael Wiese Productions.

34. Languenou, E., Benhamou, F., Goualard, F. and Christie, M. (1998) The Virtual Cameraman: An Interval Constraint Based Approach. *Constraint Techniques for Artistic Applications (Post ECAI'98 Workshop)*, Brighton, UK, August, 1-12.

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

36. Mascelli, J. (1998) *The Five C's of Cinematography: Motion Picture Filming Techniques*, Silman-James Press, USA.

37. Olivier, P., Halper, N., Pickering, J. and Luna, P. (1999) Visual Composition as Optimization. *AISB Symposium on AI and Creativity in Entertainment and Visual Art*, Edinburgh, 22-30.

38. Pickering, J. H. (2002) Intelligent Camera Planning for Computer Graphics. *PhD Thesis*, Department of Computer Science, University of York. http://www.cs.york.ac.uk/ftpdir/reports/YCST-2003-02.pdf, ftp.cs.york.ac.uk/ftpdir/reports/YCST-2003-02.pdf.

39. Seligmann, D. D. (1993) Interactive Intent-Based Illustrations: Visual Language for 3D Worlds. *PhD Thesis*, Department of Computer Science, Columbia University.

40. Thompson, R. (1998) *Grammar of the Shot*, Focal Press. An Imprint of Elsevier, 1999, ISBN 0-240-51398-3.

41. Tomlinson, B., Blumberg, B. and Nain, D. (2000) Expressive Autonomous Cinematography for Interactive Virtual Environments. In Carles Sierra, Maria Gini, and Jefrey S. Rosenschein, eds., *Proceedings of the Fourth International Conference on Autonomous Agents*, Barcelona, Catalonia, Spain, ACM Press, 317-324.

Vita Auctoris

Name: Md. Shafiul Alam

Present Address:

Permanent Address: