Intelligent Multi-Shot Visualization Interfaces for Dynamic 3D Worlds

William H. Bares **James C. Lester**
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206 USA
+1 919 515 7534
{whbares@cacs.usl.edu, lester@eos.ncsu.edu}

ABSTRACT
In next-generation virtual 3D simulation, training, and entertainment environments, intelligent visualization interfaces must respond to user-specified viewing requests so users can follow salient points of the action and monitor the relative locations of objects. Users should be able to indicate which object(s) to view, how each should be viewed, cinematic style and pace, and how to respond when a single satisfactory view is not possible. When constraints fail, weak constraints can be relaxed or multi-shot solutions can be displayed in sequence or as composite shots with simultaneous viewports. To address these issues, we have developed CONSTRAINTCAM, a real-time camera visualization interface for dynamic 3D worlds. It has been studied in an interactive testbed in which users can issue viewing goals to monitor multiple autonomous characters navigating through a virtual cityscape. CONSTRAINTCAM's real-time performance in this testbed is encouraging.

KEYWORDS
Intelligent 3D visualization, adaptive and customizable user interfaces.

INTRODUCTION
The recent emergence of high-end 3D graphics technologies offers great promise for a new generation of interactive 3D entertainment, education, and simulation-based training systems. In these and future systems, the ability to dynamically create visualizations that effectively respond to users' viewing requests, regardless of the complexity of the environment or its dynamics, is essential. To this end, recent projects have explored techniques for generating 3D illustrations [8, 9, 18, 19], producing 3D animated movies [5, 14], generating 3D animated explanations to achieve communicative goals [1, 2, 4, 13], visualizing museum walkthroughs and virtual chatrooms [6, 12], and generating 3D scenes with simulated humans for ergonomic simulation [16] and VR training [17]. Automatic camera control assistants can keep a goal object in clear view or help navigate a terrain [10, 11, 15, 16].

Intelligent visualization interfaces are critical for real-time interactive visualization of dynamic 3D worlds. For example, in next-generation 3D interactive fiction systems, multiple autonomous characters will inhabit complex environments. Autonomous characters will unpredictably navigate and interact with one another and environmental artifacts. At any time, the viewer may wish to monitor various aspects of the action or gauge the relative locations of the characters. Users of interactive 3D simulations and training applications could also post viewing goals to either monitor the environment or assist in performing tasks in the environment. Consequently, the intelligent visualization interface should, in general, attempt to satisfy viewer requests in real-time regardless of how dynamic or complex the environment. This entails continuously planning occlusion-free camera placements to view the salient features of the relevant subjects so that the viewer can immediately comprehend the subjects and their interactions with one another and the environment.

To address these issues, we have developed an intelligent visualization interface that employs a partial constraint-based framework. By reasoning from a kind of "cinematic first principles" of scene geometry, the visualization planner can solve viewers' goals to visualize given subjects. Each subject can include one or more user-specified viewing constraints such as vantage angle or distance, in addition to avoiding occlusions. Moreover, by employing a partial constraint satisfaction approach, it can provide effective alternate visualization solutions when constraints cannot be completely satisfied. The visualization interface can relax weak failed constraints and, if necessary, decompose a viewing goal into multiple shots, which can be presented as either a sequence of views or a composite view with simultaneous multiple viewports. Clarity is improved by applying several visual cues including color-coded highlights. In a real-time 3D interactive virtual environment, the user can at any time specify which subjects to view, the vantage constraints, and

preferences to control the use of inset viewports, highlighting effects, camera pace, and cinematic style.

This framework has been implemented in CONSTRAINTCAM, an intelligent visualization interface for interactive 3D worlds. Its behavior has been studied in an interactive testbed with multiple autonomous characters interacting in a dense cityscape of potentially occluding buildings. The user monitors the action by posting viewing requests to observe a specified subset of the characters as they exhibit both goal-directed and stochastic behaviors. Three autonomous characters, a police officer, Jake, and Sam wander the streets searching and competing for a lost bag of money. CONSTRAINTCAM's real-time performance and the results of an informal focus group study with viewers have yielded encouraging results.

AUTOMATED 3D CAMERA PLANNING

An intelligent 3D visualization interface requires an automated camera planner to position the camera to view the scene. Automatically planning camera shots requires solving precisely the same sets of problems that are faced by cinematographers. Human cinematographers can compose camera placements, or shots, to capture complex, live events taking place in real-world physical settings frequently beyond their control. Likewise, automated camera planners must reason about the given viewing request in the context of extant structures of the virtual environment rather than either ignoring them or making simplifying changes to them. Consequently, a general-purpose solution to the automated camera planning problem should satisfy the following four requirements:

- *User-Specified Viewing Goals:* At any time, viewers may wish to view any set of subjects, and they may do so by stipulating specific viewing constraints on each.

- *Real-time Behavior Unpredictability:* Events may occur in an unpredictable fashion resulting from interactions between autonomous characters, simulation events, and interactive user manipulations.

- *Environmental Complexity:* Worlds are populated by objects arranged in non-trivial configurations.

- *World Non-Interference:* The world should not be modified to simplify the visualization problem.

Prior works on automated camera planning do not provide a general-purpose solution to address these requirements. One family of systems employs camera positions pre-specified relative to the subject(s) being viewed [1, 2, 8, 9, 18, 19]. This approach fails when the camera must view arbitrary combinations of subjects with specific constraints, or when unanticipated motion or obstructions occlude the subjects of interest. IBIS can overcome viewing failures by using multi-view illustrations and cutaways of occluding obstructions, and CATHI has a facility for transparency [4, 9]. Though not a focus of this work, transparency and cutaways could be used in special cases such as when a subject is enclosed within another object.

Idiom-based systems encode knowledge of cinematography to sequence shots of commonly occurring actions [4, 5, 12, 13, 14]. Idioms focus on the complementary issue of sequencing shots rather than solving camera positions. Existing idiom-based systems use a finite set of pre-specified relative camera placements in lieu of camera placement solvers. Thus, these idiom-based systems can often fail to find acceptable shots when subjects occupy unanticipated relative spatial configurations, or when structures in the world occlude the subjects of interest.

In contrast, the constraint satisfaction approach to automated camera planning casts viewing goals as constraint satisfaction problems. When a viewer issues a request to visualize particular subjects of interest and specifies how each should be viewed, a constraint solver attempts to find a solution camera placement. CAMDROID supports a broad and powerful set of camera constraints, but employs a numerical constraint solver that is subject to local minima failures [6, 7]. In our prior work, a task model was used to direct a real-time constraint-based camera planner to provide views of a learner's avatar [3]. Neither of these two constraint-based systems provided a systematic solution for handling constraint failures.

Automatic camera control assistants vary camera position to avoid occlusions of a goal object or satisfy screen-space constraints on how subjects appear on-screen [10, 16]. Automated camera navigation assistants adjust camera motion speed based on distance to the target or guide the camera along specified optimal vantages as a user navigates over a terrain [11, 15]. Neither automated viewing nor navigation assistants can address user-specified viewing goals because they focus on specific subsets of these issues or on controlling relatively low-level parameters and frequently require considerable user inputs.

3D VISUALIZATION INTERFACE

We have developed a flexible 3D cinematic visualization interface that allows viewers to at any time select which subject(s) to be viewed, specify the vantage constraints for each subject, select cinematic style or pace, choose the style of highlights, and control the amount of information displayed at one time on the screen.

Viewers may begin by selecting which subset of the principles to view. For example, in the interactive testbed, they may select from the set of {Cop, Jake, Sam, money bag, bank, or the hideout Joe's Place} by depressing the toggle buttons along the bottom edge of the screen. They may then use a dialog box to specify the desired optimal viewing vantage angle for each subject. Users submit a viewing goal by pressing the *Ask* button.

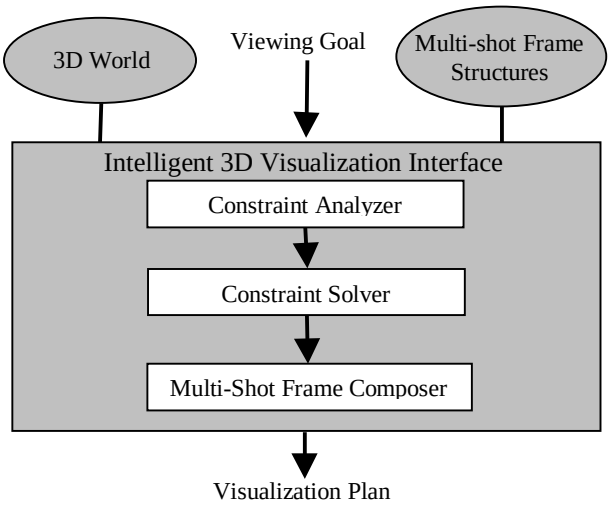


Figure 1: CONSTRAINTCAM architecture

Users can also specify the information content per screen, which influences the number of simultaneous inset viewports used to present multi-shot solutions.

The user can also indicate if *highlight effects* should be used and if so what graphical style should be used.

- *Outline:* Draw a colored rectangle around the object.
- *Bounds:* Draw a 3D bounding box around the object.
- *Color Tint:* the object is tinted by the highlight color.
- *Pulsate:* The color of the highlight blinks if this option is selected.

The user may specify the *cinematic pace* by selecting from the following options:

- *Slow:* camera moves slowly and shots have long duration.
- *Medium:* camera moves at a moderate speed and shots have a moderate duration of approximately 5 seconds.
- *Fast:* camera travels rapidly and shots have short duration of approximately 3 seconds.

A user can select a *cinematic style* by choosing one of the following strategies for sequencing camera shots.

- *Informative:* the camera is positioned to view the subject(s) from establishing shots that view the subject(s) from the user's specified preferred vantage angle. The camera will move to track the subject(s).
- *Mixed:* Selects a variety of shot sequences ranging from establishing views, pan shots, and ease-in/out.
- *Dramatic:* Camera shot sequences involving sweep-arounds, ease-ins, and ease-outs are employed.

The stylistic sequences of camera shots are composed of one or more shots of the following types presented in sequence. The selected *cinematic style* dictates which specific sequences are instantiated.

- *Establishing shot:* film all subjects from medium or long range to show their relative locations or attributes.
- *Pan:* Fix camera position and rotate camera to track subject(s).
- *Ease-in:* Begin at establishing shot, then move in for a close-up.
- *Ease-out:* Opposite of ease-in.

PARTIAL CONSTRAINT CAMERA PLANNING

Camera planning begins when a viewer posts a 3D visualization request. For example, she might request to view three subjects, two of which happen to be separated by a great distance, with the objective of comparing their relative locations and physical attributes. This visualization request implies the following constraints:

(1) All subjects are visible in a single view to establish their relative locations and attributes

(2) The camera must be sufficiently near the subjects that their distinguishing attributes are recognizable

(3) Each subject should be viewed from an angle that reveals its distinguishing features

(4) Occluding obstacles in the environment cannot obscure the subjects of interest.

Such visualization requests are solved by CONSTRAINTCAM whose primary modules are illustrated in Figure 1. Knowledge sources include the description of the 3D world, a library of multi-shot frame structures, and the given viewing constraint problem. Principle modules include the Constraint Analyzer, Constraint Solver, and Multi-Shot Frame Composer. These computations result in a hierarchical visualization plan of shots and insets.

Given a viewing goal, CONSTRAINTCAM attempts to find a camera placement that will satisfy all of the constraints. The Constraint Analyzer identifies regions of space in which to place the camera to satisfy each constraint. If a solution is possible, it is found by computing a common region of space in which to place the camera so that all of the constraints are satisfied. If a solution in the form of a single shot cannot be found, then the Constraint Solver identifies which pairs of constraints are incompatible. The weakest incompatible constraints are relaxed to permit a solution, or the problem is decomposed to form a multi-shot solution. The Multi-Shot Frame Composer exploits a repository of multi-shot frame structures to create sequential or composite visualizations of the multiple shot solution. The solution is expanded into a hierarchical visualization plan, whose child nodes represent individual shots, and coded arcs denote sequential or simultaneous display of child shots in insets. The plan is then rendered in real-time and typically lasts for a duration of at least four seconds, then the entire 3D visualization planning process is repeated to reflect new developments in the environment.

Formulating Camera Constraints

A user's visualization goal is expressed as a constraint problem consisting of a number of constraints on the subject(s) to be viewed. The Constraint Solver supports four types of constraints (a subset of Drucker's constraints [6, 7]). Each constraint can be applied to any subject to be viewed by the camera and includes a relative priority and a marker indicating whether that constraint can be relaxed.

- *Vantage Angle:* Indicates the permissible range of relative orientations between the camera and the subject, e.g., which faces(s) of an object the camera is allowed to view, and of these, which is the optimal.
- *Viewing Distance:* Specifies the minimum and maximum allowable distances between the camera and the subject along with the optimal distance.
- *Occlusion Avoidance:* Dictates that camera position should be adjusted when necessary to prevent the subject from being occluded by obstacles.
- *Room* Enclosure: Limits the camera position to remain within an optional enclosed rectangular region.

The strength of a constraint is determined by the product of the priority of the subject it applies to and the given priority of that constraint. During relaxation, constraints of lower strength will be disabled before those of greater strength.

Analyzing Consistent Regions

For each given constraint, the Constraint Analyzer must determine a *consistent region* of space within which the camera must be placed to satisfy that constraint. Consistent regions are expressed in spherical coordinates and packed into discrete bit arrays (each bit represents a 3º x 3º span of spherical space) so that fast register bit-wise operators can compute the solution intersection in constant-time. Each consistent region is expressed in terms of a local spherical coordinates system with origin at the center of the subject upon which that constraint is applied.

For example, the consistent region satisfying the occlusion avoidance constraint for a subject *S* is found by projecting the bounding boxes of nearby potentially occluding obstacles onto a sphere surrounding the subject *S* (Others have used a similar method for finding occlusions [7, 16]). These projections are then converted into a global spherical coordinate system and then negated to represent occlusion-free regions of space for viewing subject *S*.

To compute an intersection of the consistent regions, each local consistent region must be first converted into a common global spherical coordinate system. This global spherical coordinate system is defined so that its origin is at the center of all subjects of interest. Each local spherical coordinates consistent region is projected onto the surface of the surrounding global sphere to affect the conversion to global spherical coordinates.

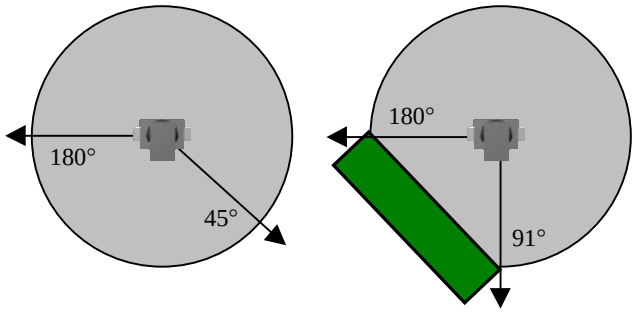
Computing Constraint Solutions

Once the consistent regions for all constraints have been computed and converted into a common global representation, they are all intersected. If the cumulative intersection region *R* is non-empty, then the Constraint Solver searches for the spherical coordinates point (θ, ϕ) within *R* that is nearest the optimal vantage for viewing the set of subjects. The optimal vantage for the set of subjects is found by evaluating one of several functions of the optimal vantage of each individual subject, e.g., average or vantage of highest priority subject. The camera distance *dist* away from the subject(s) is computed via intersecting minimum and maximum distance intervals corresponding to the consistent regions for the viewing distance constraints. If the optimal vantage is occluded, then *dist* may be decreased to put the camera in front of the nearest obstacle found by ray casting against obstacle bounding volumes. Point (θ, ϕ, *dist*) is converted from spherical coordinates to Cartesian coordinates to set the position of the camera which is aimed at the center of the subject(s).

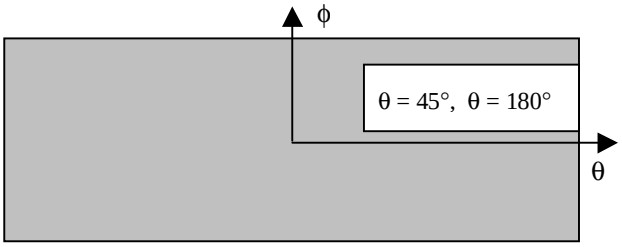
For example, assume that vantage angle and occlusion-avoidance constraints are applied to view the subject shown in Figure 2. Figure 2(a) illustrates the horizontal component of the allowable vantage constraint which requests a front sided view of the Cop θ [45°, 180°]. Figure 2(b) shows the region of space marked inconsistent with respect to the occlusion-avoidance constraint since it includes an obstacle blocking vantages θ [91°, 180°]. Figure 2(c) plots the spherical consistent region for the vantage constraint where consistent vantages (θ, ϕ) are marked in white. Figure 2(d) plots the occlusion-avoidance consistent region with the occluded region marked in dark gray (inconsistent). For the sake of illustration, the ϕ (elevation) dimension is assigned arbitrary values since the example only presents a "top-down" view. Figure 2(e) plots the consistent region computed from the intersection of the vantage and occlusion-avoidance constraint consistent regions. To satisfy both constraints, the camera vantage component θ must lie within [45°, 90°]. The elevation component assumes the arbitrary range of ϕ [15°, 60°]. Assume that the optimal vantage angle was (θ 135°, ϕ 45°). The solution consistent region is searched beginning from (θ 135°, ϕ 45°) to find the consistent vantage nearest to the optimal, which in this example happens to be (θ 90°, ϕ 45°). This places the camera to view the front of the Cop as shown in Figures 2 (f) and (g).

Constraint Failure Heuristics

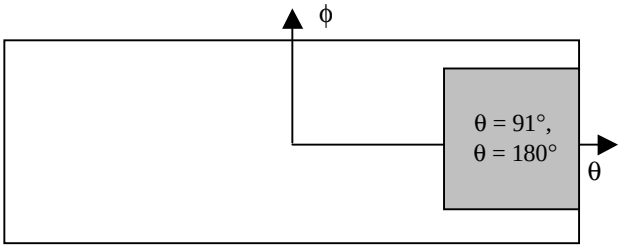
If no solution can be found for the given constraints, then the Constraint Solver attempts to find an alternate solution. For many interactive 3D applications, it is acceptable—frequently it is even critical—for the automated camera to present some view of the scene that gives the viewer some,



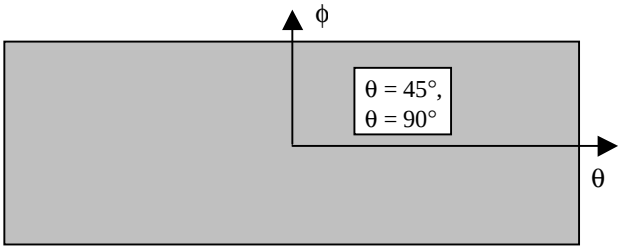
(a) Allowable vantage region (b) Occluded region



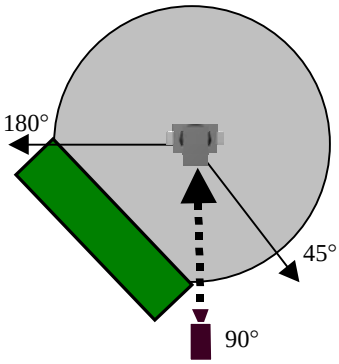
(c) Vantage consistent region



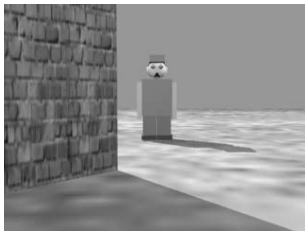
(d) Occlusion consistent region



(e) Constraint solution consistent region



(f) Solution camera placement



(g) Camera view

Figure 2: Camera constraint solution example

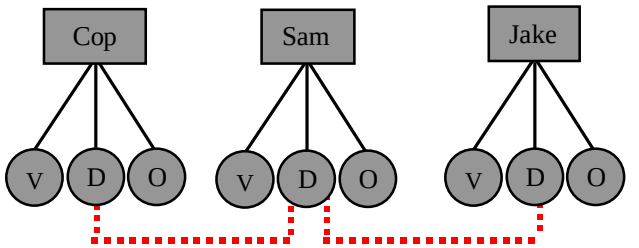
if not all, of the information requested. To accomplish this, the Constraint Solver first identifies the combinations of constraints that are incompatible. It next attempts to find a maximal solution satisfying as many of the higher strength constraints as possible.

Combinations of incompatible constraints are identified by constructing an *incompatible constraints pair graph.* The Constraint Solver creates a node for each $C_{s,i,}$ the i^{th} constraint on subject S. Next, it adds an arc connecting nodes { $C_{s1,i1}$, $C_{s2,i2}$ } if their consistent regions $R_{s1,i1}$ and $R_{s2,i2}$ fail to intersect.

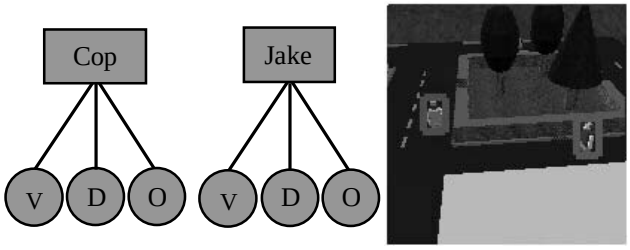
The Constraint Solver then repeatedly relaxes weaker constraints until no incompatible constraint pairs remain. Incompatible constraints are tested for relaxation in order from lowest to highest strength. When a constraint can be relaxed, it deletes all incompatible constraint pair graph arcs that involve that constraint. It continues until either no more constraints remain to be tested or no more arcs remain in the graph. If no incompatible constraint pairs arcs remain, then it submits the resulting relaxed constraint problem to the constraint solver. If relaxation was successful, it returns a single shot camera solution to the relaxed problem. For example, assume the vantage constraints of subject S_1 and S_2 are incompatible and the vantage constraint of S_2 is of lower strength. The vantage constraint of S_2 is then relaxed, and the arcs dependent upon it are deleted. Since no incompatible constraint pairs remain after relaxation of S_2's vantage constraint, then the relaxed constraint problem can be solved.

If relaxation is not possible, then the Constraint Solver can decompose the viewing constraint problem into a multi-shot solution. The Constraint Solver attempts to satisfy as many constraints as possible in each sub-problem to minimize the number of shots. It places each incompatible constraint of the pair { $C_{s1,i1}$, $C_{s2,i2}$ } into a distinct sub-problem P_k. It then inserts all possible compatible constraints into each sub-problem. Thus for sub-problem P_k including constraint $C_{s1,i1}$, it adds constraint $C_{s,i}$ if no incompatible constraint pairs arc joins these constraints. This maximal heuristic attempts to include as many subjects as possible in each sub-problem (shot) which helps establish the relative location attributes. Finally, each sub-problem is solved to produce a camera shot.

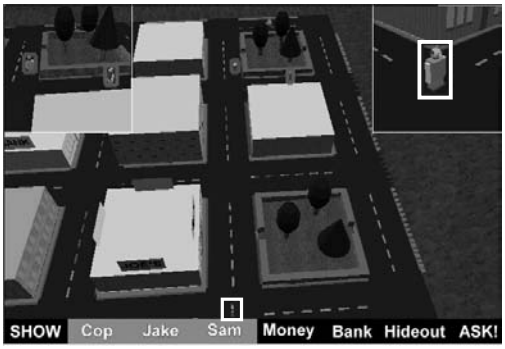
For example, Figure 3(a) depicts an incompatible constraints pair graph for three subjects Cop, Sam, and Jake. *V* indicates vantage constraints, *D* distance constraints, and *O* occlusion constraints. Dotted arcs join the incompatible distance constraints. Sam (lower middle of Figure 3(c)) is too far away from the Cop and Jake (upper middle) for all three to appear in a single view from a distance near enough so each is recognizable. Since there are no incompatible constraint pair arcs between the Cop and Jake, then the Constraint Solver can compute a shot of



(a) Incompatible constraint pairs graph with failed constraints



(b) Decomposed shot of nearby characters Cop and Jake



(c) Overview with decomposed shots in inset viewports

Figure 3: Example constraint decomposition

both characters (Figure 3(b)). The decomposed shot depicting Sam appears in the top right inset of Figure 3(c).

Multi-Shot Solutions

If the Constraint Solver has decomposed a constraint problem, then the Multi-Shot Frame Composer determines how to present the multiple camera shots. Composite shots combine a main viewport with one or more inset viewports allowing the viewer to gauge relationships between subjects. Alternatively, the user may prefer to focus on each shot individually in sequence. The Multi-Shot Frame Composer draws from a repository of multi-shot frame structures, each of which displays from one to four camera shots per screen. Figure 3(c) illustrates the display for a preference of two insets to display the two decomposed shots along with an overview shot of all three characters found by relaxing the failed distance constraints.

Coordinating Multi-Shot Visualizations

To improve the clarity of the resulting visuals methods are employed to establish connections between subjects appearing in insets, avoid use of redundant inset shots, and highlight the subjects of interest. When a multiple shot solution is produced, constraints are relaxed to create a supplemental overview shot, which includes all subjects of the viewing goal. This overview shot is presented in the main viewport in conjunction with the decomposed shots in the inset(s) to help connect the subjects in insets to their respective locations in the overview shot.

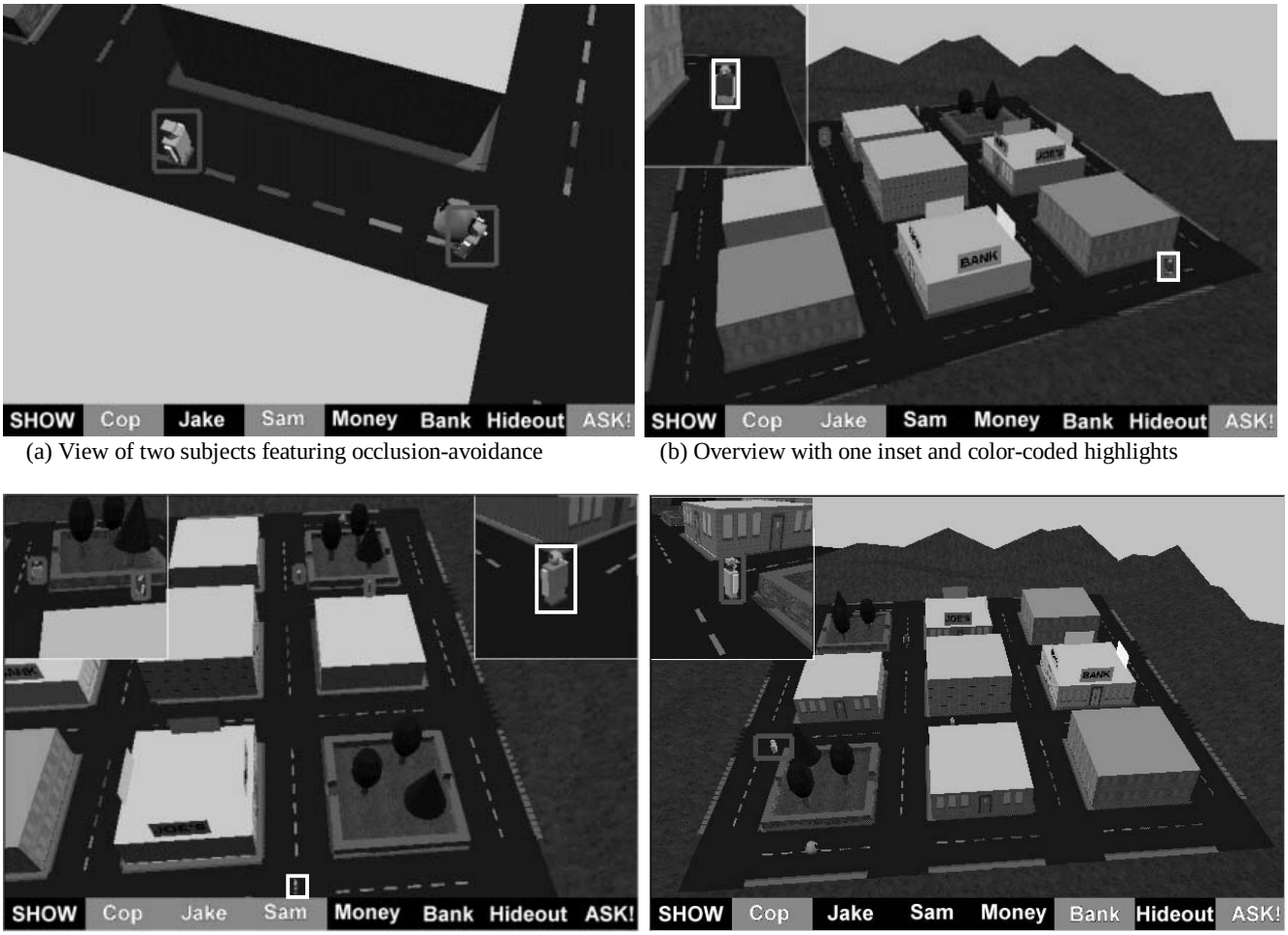
The resulting inset shots are compared to the overview shot to cull redundant inset shots which do not depict subject(s) of interest significantly better according to a measure of constraint satisfaction success than the overview shot. Each camera shot can be evaluated in terms of how well the constraints on the subject(s) shown are satisfied. A constraint satisfaction success rating is computed for each shot by deducting a weighted penalty function for each failed constraint. An optimality rating can be computed for each shot based on how closely the camera is placed relative to the desired optimal vantage for viewing each subject. Highlights in the form of outline rectangles, bounding boxes, or blinking colors can be applied to draw attention to subjects. Color-coded highlights are used to tie subjects in the insets to their positions in the overview shot. Highlights are also applied when a subject's on-screen size is below a threshold or the subject is occluded as indicated by the constraint satisfaction success evaluation.

EXAMPLE INTERACTION

This example illustrates how CONSTRAINTCAM[1] responds to a series of viewing goals in the interactive 3D testbed. The user posts a viewing goal by selecting which subject(s) are to be viewed using labeled toggle buttons then pressing the *Ask* button. In this example, viewing goal constraints are specified to disallow relaxation so that failures exercise multi-shot solutions with relaxed constraint shots in main viewport overview shots. Initially, the viewer allows at most one inset viewport and wishes to view the Cop and Sam. The resulting camera view in Figure 4 (a) depicts Sam to the right of the police officer with both surrounded by buildings on either side. Note that the camera has elevated its position to obtain an occlusion-free view. The camera vantage slightly in front of and to the right of the characters is the result of averaging the user-specified optimal vantages of each character (rear for the police officer, and front-right for Sam) in addition to elevation of the camera to avoid the occluding building.

The viewer next requests a view of the Cop and Jake. In Figure 4(b) the camera cannot find a clear shot of both since they are located on opposite sides of town. Distance constraints fail since a single shot of both characters would

[1] The interactive 3D testbed consist of approximately 55,000 lines of C++ code and achieves frame rates of approximately 7 to 15 frames per second on a 333 MHz Pentium II with a Permedia2 OpenGL accelerator.



(a) View of two subjects featuring occlusion-avoidance

(b) Overview with one inset and color-coded highlights

(c) Overview with two multi-shot insets

(d) Overview and inset of cop but culled multi-shot of bank

Figure 4: Example screen shots

require the camera to be placed too far away for either to be recognizable. Multi-shot decomposition results in one shot of the Cop and a shot of Jake. Since one inset is allowed by the viewer, the Multi-Shot Frame Composer creates a relaxed-constraint overview shot in the main window and uses the inset viewport to first show the detail shot of Jake. Jake, outlined in white in the overview and inset, stands at the lower rightmost street corner. Seconds later the inset will present the shot of the Cop, outlined in red, who appears in the upper left of the overview shot.

Next, the viewer allows up to two insets. A viewing goal is posted to show the Cop, Sam, and Jake. As Sam moves too far away, the three characters can no longer clearly appear in a single view, and a multi-shot solution consisting of an overview shot and two insets is employed as shown in Figure 4 (c). The first shot, which appears in the upper left inset, satisfies the constraints on the Cop and Jake. The second shot of the decomposition (top right inset) depicts Sam who is outlined in white and stands in the bottom center of the overview shot. For additional details on how the partial constraint solution was computed in this multi-shot example, you may refer back to Figure 3.

Lastly, the viewer shifts focus to the Cop and bank. The multi-shot solution presents the Cop (outlined in red) in an inset viewport since he appears small in the left side of the overview shot, but culls out the multi-shot of the bank since it appears clearly in the middle right of Figure 4(d).

CONCLUSIONS AND FUTURE WORK

As interactive 3D worlds appear in an expanding range of entertainment, educational, and training systems, they place an increasingly heavy demand on real-time visualization systems to respond to user-specified viewing requests in complex environments. We have proposed an intelligent visualization interface framework for planning goal-directed shots in interactive 3D worlds. By exploiting the flexibility of partial constraint satisfaction, it can compute "next-best" solutions to difficult viewing problems by relaxing less critical constraints and by creating multi-shot solutions with customized highlights and insets. While this work addresses many of the core issues in automated camera planning, much remains to be done. For example, a direct-manipulation interface could be used to select the subject(s) and specify any of viewing constraint minimum, maximum, and optimal values. The silhouettes, rather than

bounding boxes, of occluding objects should be plotted to handle irregular and concave polyhedral objects. Presently, the algorithm misses some solutions that place the camera in front of an occluding object. We are developing a more complete representation of distances of potential occluding objects. We will be exploring these issues in ongoing and future research.

ACKNOWLEDGEMENTS

Thanks to: the multimedia design team (Tim Buie, Mike Cuales, Patrick FitzGerald, Alex Levy, and Rob Gray) for cinematography suggestions and video production. Support is provided by a grant from the National Science Foundation (Faculty Early Career Development Award IRI-9701503), the North Carolina State University IntelliMedia Initiative, and an industrial gift from Novell.

REFERENCES

1. Elisabeth André, W. Finkler, W. Graf, T. Rist, A. Schauder, and W. Walhster. WIP: The automatic synthesis of multimodal presentations. In M.T. Maybury, editor, *Intelligent Multimedia Interfaces*, chapter 3, AAAI Press, 1993.

2. William H. Bares and James C. Lester. Realtime generation of customized 3D animated explanations for knowledge-based learning environments. In *AAAI-97: Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 347-354, 1997.

3. William H. Bares, Luke S. Zettlemoyer, Dennis W. Rodriguez, and James C. Lester. Task-Sensitive Cinematography Interfaces for Interactive 3D Learning Environments. In *IUI-98: Proceedings of the 1998 International Conference on Intelligent User Interfaces*, pages 81-88, 1998.

4. Andreas Butz. Anymation with CATHI. In *Proceedings of the Ninth Innovative Applications of Artificial Intelligence Conference*, pages 957-962, 1997.

5. David B. Christianson, Sean E. Anderson, Li-wei He, David H. Salesin, Daniel S. Weld, and Michael F. Cohen. Declarative camera control for automatic cinematography. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 148-155, 1996.

6. Steven Drucker and David Zeltzer. Intelligent camera control in a virtual environment. In *Proceedings of Graphics Interface '94*, pages 190-199, 1994.

7. Steven Drucker and David Zeltzer. CamDroid: A system for implementing intelligent camera control. In *Proceedings of the 1995 Symposium on Interactive 3D Graphics*, pages 139-144, 1995.

8. Steven Feiner. APEX: An experiment in the automated creation of pictorial explanations. *IEEE Computer Graphics & Applications*, pages 29-37, November 1985.

9. Steven Feiner and Dorée Seligmann. Cutaways and ghosting: satisfying visibility constraints in dynamic 3D illustrations. *The Visual Computer*, pages 292-302, August 1992.

10. Michael Gleicher and Andrew Witkin. Through-the-lens camera control. In *Proceedings of ACM SIGGRAPH '92*, volume 26, pages 331-340, July 1992.

11. Andrew Hanson and Eric Wernert. Constrained 3D navigation with 2D controllers. In *Proceedings of IEEE Visualization '97*, pages 175-182, 1997.

12. Li-wei He, Michael Cohen, and David Salesin. The virtual cinematographer: A paradigm for automatic real-time camera control and directing. In *Proceedings of ACM SIGGRAPH '96*, pages 217-224, 1996.

13. Peter Karp and Steven Feiner. Issues in the automated generation of animated presentations. In Proceedings of Graphics Interface '90, pages 39-48, 1991.

14. Peter Karp and Steven Feiner. Automated presentation planning of animation using task decomposition with heuristic reasoning. In *Proceedings of Graphics Interface '93*, pages 118-126, 1993.

15. J. Mackinlay, S. Card, and G. Robertson. Rapid controlled movement through a virtual 3D workspace. In *Proceedings of ACM SIGGRAPH '90*, pages 171-176, 1990.

16. Cary B. Phillips, Norman Badler, and John Granieri. Automatic viewing control for 3D direct manipulation. In *Proceedings of the 1992 Symposium on Interactive 3D Graphics*, pages 71-74, 1992.

17. Jeff Rickel and Lewis Johnson. Integrating pedagogical capabilities in a virtual environment agent. In *Proceedings of the First International Conference on Autonomous Agents*, pages 30-38, 1997.

18. Dorée Seligmann and Steven Feiner. Automated generation of intent-based 3D illustrations. *Computer Graphics*, 25(4), pages 123-132, 1991.

19. Dorée Seligmann and Steven Feiner. Supporting interactivity in automated 3D illustrations. In *Proceedings of Intelligent User Interfaces, '93*, pages 37-44, 1993.