An Architecture Based on Constraints for Augmented Shared Workspaces

CARLOS HENRIQUE QUARTUCCI FORSTER
CLÉSIO L. TOZZI
Dept. of Computer Engineering and Industrial Automation - DCA
School of Electrical and Computer Engineering - FEEC
State University of Campinas - UNICAMP
C.P. 6101, 13083-970 - Campinas, SP, Brazil

{forster,clesio}@dca.fee.unicamp.br

Abstract – Augmented shared workspaces are an instance of augmented reality which move the collaborative work from the desktop to the real workplace, enabling higher interaction level with coworkers and allowing implementation of advanced techniques of human-machine interfaces. In this paper, we advocate the employment of a constraint system as the document model (or program) for the implementation of augmented reality applications. We show that constraint technology is synergistic with augmented reality and collaborative work in many aspects. An architecture model based on agents and constraints is then proposed. Our point of view is supported by the observation of this technology being employed in Computer Graphics and Human-Computer Interaction research and the implementation of typical cases.

Keywords – Augmented reality, constraint systems, constraint logic programming, shared workspaces, CSCW, multimedia document, agent theory.

1 Introduction

In near future, as computer technology advances, tasks executed by humans away from the desktop will gain computer support. An enabling technology for this accomplishment is Augmented Reality (AR). AR is a consequence of computer vision algorithms, realistic rendering and advanced human-machine interfaces. In AR, information can be displayed for the worker directly in his workplace due to special glasses, projectors or displays. AR is adequate for computer supported cooperative work (CSCW). For instance, a computer system can monitor the group of workers, anticipating their actions, enabling communication among them and minimizing the effect of remoteness, instruct them, warn them of mistakes and dangerous situations and inform them what someone else is doing. The same vision algorithms used for the reality augmentation can be used for cooperative work support.

In this paper, we intend to show that using constraints to represent AR applications is very appropriate and convenient. Constraint-based systems have been used since the earlier days of computer graphics and vision research. Constraints have been used in graphics for automatic construction of geometric models. Constraint propagation has been used in vision as a solution to the edge labeling problem [Winston, 92]. Constraint systems are recommended for expressiveness and difficult combinatorial problems such as scheduling. They are also used for fast prototyping. Recently, constraints have earned more attention from researches due to the higher processing power of hardware and the benefit of short development time. The main advantages of using constraint technology for modeling AR applications are expressiveness, parallelism, incremental solution, real time efficiency, ability to model geometric relations, ability to model human-computer interface and ability to model space and time dependencies.

This paper is organized as follows: in Section 2 we discuss AR applied to shared workspaces; in Section 3 we briefly introduce constraint-based systems and show the usefulness of these systems in our context; in Section 4 an architecture for augmented shared workspaces is proposed; in Section 5, we present examples and results and in Section 6 we present the conclusions.

2 Augmented shared workspaces

Augmented shared workspaces is a promising technology for computer assisted work in near future. CSCW systems used to be confined to desktop environments. AR systems allow computer supported collaborative work away from the desktop. Users can perform their manufacture and maintenance tasks in real world, and the computer system can assist them with additional information and augmented perception. Computer assistance can be inserted in this context in a natural way, as

1530-1834/01 $10.00 © 2001 IEEE

the discontinuity imposed to the work routine is minimal. A good example of an Augmented Reality collaborative workspace is the Studierstube project [Schmalstieg et al., 00].

2.1 Augmented reality

Augmented reality is a modality of virtual environment where the perception of the real world is not blocked [Azuma, 97]. Instead, additional information represented by virtual objects is superposed to the perception. On line AR can be accomplished through the use of special see-through head mounted displays (HMD). There are two kinds of display technology involved: video see-through (VST) and optical see-through (OST). VST consists of a set (pair) of cameras and a pair of displays meant for each eye. The real world is captured by the cameras, rendering of virtual objects is performed and composed with the real world perception stereo video and, finally, the composition is displayed for the user. OST displays employ semi-transparent lenses to perform the composition. Input video can still be used, but it is not necessary. The rendering technique must consider that real world perception will not be erased. Off line AR is a special case of VST without the real time restriction.

Common applications of AR are visualization, annotation, manufacture, maintenance, robot trajectory planning and entertainment. More recently, the collaborative potential of AR has been discovered.

AR demands computer vision algorithms for correct alignment of virtual objects in the real world, for keeping control of real objects so as to allow interpretation of events and for depth estimation in order to represent accurately occlusion of real objects by virtual objects and vice-versa. An important problem is sensor fusion, as current systems tend to adopt hybrid mechanisms for tracking instead of relying on just one type of device. AR system designers should consider multimedia and human-computer interfaces issues as well because such systems may integrate many types of media and non-conventional (non-desktop) interface devices.

The augmented environment can be inhabited by virtual agents. Agents are entities able to sense the environment and actuate on it. Agent technology has become a paradigm of software development. Autonomous software agents can interact with the augmented environment in order to present information to users in a natural manner. These agents can be embodied in virtual creatures, or avatars, such as virtual humans. An augmented shared workspace can be modeled as an environment for agents, where agents can be humans (the users), software agents (act in virtual world) and robotic agents (act in real world).

2.2 Supporting cooperative work

In order to support cooperative work, we can mimic desktop multimedia shared workspaces. The developers of these systems were concerned with the following aspects: a worker should be able to observe the action of the other participants, he should be able to initiate face to face communication with some group of participants and he should be able to scratch the tasks the group will be performing on a cooperative whiteboard [Buford, 94] [Fluckiger, 95].

Observing the action of other workers is what we call WYSIWIS (what you see is what I see) interface. The biggest problem in this case is the limited size of displays for the large amount of information to be displayed. For non-desktop workspaces, this may not be a problem as you can display the tasks performed by other workers in the 3D environment as virtual objects.

Face to face communication is perhaps the most important issue as social behavior can surpass some collaboration problems, such as access exclusion. Emotional signs are also important for clear communication. The efforts are directed to integrating conferencing systems with the workspace. In the out of the desktop case, with AR, people can see each other and communicate directly. In the telepresence case, people can be represented as avatars (virtual humans) in the augmented environment.

The design of cooperative whiteboard in virtual environments is a challenge for interface researchers. Manipulating virtual objects by dragging, moving or activating, just like a painting program, is the traditional approach. The concern, here, is the concurrency problem. A concurrency resolution policy should be adopted. The most common approaches are relying on social behavior and implicit locking. In the former case, no exclusion method is implemented, leaving the control solely to the users. In the latter case, a lock is created when a user begins his task, so that other users are not able to interfere until the user stops for a small amount of time. For augmented workspaces we should use the former policy for real objects and the latter for virtual objects. Other types of floor control are the moderator and explicit locking by request, using queues. Another possibility is the indirect manipulation of virtual objects through real world objects. An example of cooperative indirect manipulation is presented in [Baker *et al.*, 98]. This approach has the advantage of being naturally asynchronous.

Now, we move our discussion to the field of constraint systems. The relation among augmented reality, cooperative work and constraint systems will be established in section 3.3.

3 Constraint-based systems

A constraint is a relation between objects or variables. An equation or inequality represents each constraint. The constraint satisfaction problem (CSP) is the problem of finding a state that satisfies all constraints. A constraint system solves this problem, maintaining the constraints satisfied when a variable changes.

This paradigm has proven to be useful in a wide range of problems and has attracted the attention of researchers in the last few years [Barták, 99]. The term Constraint Programming (CP) is used to refer to the study of computational systems based on constraints. Constraint Logic Programming (CLP) is a generalization of logic programming, where the logic statements are also constraints.

Constraint systems firstly appeared in AI, graphics and vision research. Recently, it is closed related to Operations Research when dealing with NP-hard combinatorial problems.

There are many algorithms available for a variety of constraint problems. The constraint solver depends on the domain of the variables (boolean, finite domain, integer, interval, real), the topology of the constraint system (one-way constraints, linear multi-way constraints, presence of cycles, non-linear simultaneous constraints), policies to deal with over or under constrained systems and support for higher order constraints (conditionals) and side-effects (procedure calls).

Common algorithms are topological sort for one-way propagation, combinatorial search algorithms, arc consistency, symbolical solution of equations, degrees of freedom analysis, interval propagation, simplex based method, iterative numerical methods among others. Surveys on algorithms can be found in [Barták, 99] and [Kumar, 92]. Introductory texts are [Rossi, 99], [Wallace, 95], [Frühwirth *et al.*, 92], [Freeman-Benson, 90] and [Leller, 88]. Olsen [92] describes constraint systems in the human-computer interface point of view.

Concurrent solution for constraint satisfaction is called Concurrent Constraint Programming (cc). A framework for cc is based in multi-agent systems, see [Montanari and Rossi, 95].

Existing systems are very application specific. Most have a limited vocabulary of constraint and variable types. Those systems using iterative numeric algorithms should be more controllable so as to avoid stability problems. The implementation of a completely new system for different problems is very usual because crafting a generic constraint system is very difficult.

Among the applications for constraint systems we can cite scheduling problems (time dependency), layout design (spatial dependency), assignment problems (combinatorial search), real-time control, interfaces, multimedia orchestration, text page layout, interactive graphics and animation, vision problems (edge labeling), verification of system conditions, debugging, diagnostics, constraint queries for databases, network management, spreadsheets, interactive problem solving and computational biology problems [Rossi, 99].

3.1 A motivation towards constraint logic programming

An example from [Leler, 88] to introduce CLP is the conversion between temperature scales. While in conventional imperative programming we need 6 attribution statements to convert Celsius to Fahrenheit, Celsius to Kelvin, Kelvin to Celsius, Kelvin to Fahrenheit, Fahrenheit to Celsius and Fahrenheit to Kelvin, in CLP only the 2 statements shown in equation (1) are necessary. When one of the variables C, K or F changes, the constraint system corrects the other two in order to maintain the constraints satisfied.

$$\begin{cases} C = (F-32)\times 5/9 \\ K = C + 273 \end{cases} \quad (1)$$

This example illustrates the expressive power of the CLP approach. Another example is a program to draw a regular pentagon. Doing this in an imperative language requires a lot of calculations from the programmer. In a constraint language, the programmer needs just to specify the position of a vertex and the central point and declare that the distance between each two vertices connected by an edge is the same and the distances from vertices to the central point are equal. An ambiguity problem arises here, as a five-pointed star is also a solution that satisfies the constraints. In this case, the system is under constrained, additional constraints can solve the ambiguity problem, but can also make the system over constrained, where a solution can be hard to find.

3.2 Geometric constraint systems

Surveys of geometric constraint systems are [Badros, 98] and [Hower and Graf, 96]. Such systems should rely on non-linear constraints. To achieve efficiency, a general numerical constraint solver is often avoided. Another important characteristic is the use of a limited vocabulary of constraints. Constraint solvers intended for the layout problem and such as CSVG [Badros *et al.*, 00], Amulet [Myers, 96] and LayLab [Graf, 96] use linear one-way or multi-way constraints. In Bramble [Gleicher, 93], an incremental numerical non-linear constraint solver is employed, considering continuous movement of interface objects. GLIDE [Ryall *et al.*, 96] is a graph visualization system based in spring simulation for over constrained declarations. In Chimera [Kurlander, 93] the Levenberg-Marquadt algorithm (variation of Newton-Raphson) is employed to find the solution of the non-linear system. The constraint vocabulary of Chimera is very interesting as absolute and relative constraints are considered. QOCA [Helm *et al.*, 95] employs quadratic optimization, which is useful for constraints defined by Euclidean distance between points. Diehl and Keller [00] implement constraints and some application scenarios in VRML. Sced [Chenney, 93] is a geometric editor supporting one-way non-cyclical constraints. DLoVe [Deligiannidis, 00] is a distributed VR-based CSCW system that employs a one-way constraint solver.

Numerical solvers based on Newton-Raphson algorithm may not be efficient enough and requires an initial guess. Other approaches are possible, such as interval methods [Benhamou, 95], degrees of freedom analysis [Kramer, 92] and symbolic methods (NP-hard). The constraint system can be modeled as a set of differential equations whose asymptotic equilibrium state satisfies the constraints as described in [Witkin *et al.*, 87, 88] for animation control. The incremental nature of the problem allows efficient implementation.

3.3 Synergistic advantages

We list some advantages of implementing AR applications over a constraint-based system instead of conventional imperative and event-based models. The general advantages of choosing a constraint based implementation is the expressiveness of constraint languages, their flexibility, ease of use, ease of learning and fast prototyping. Expressiveness means that the program description is closer to the problem description instead of the solution description.

Flexibility is the ability to use the system in multiple contexts. Ease of use and ease of learning mean that the languages can be designed to be easily used and learned, with a limited vocabulary of variable types and constraint types and visual representation (see [Ryall *et al.*, 96]). Fast prototyping is the ability to shorten development time. Parallelism and real time efficiency are desirable points for an on line AR system. The incremental solution characteristic of the constraint satisfaction guarantees that the implemented algorithms are incremental and provides a solution each time it is necessary, i.e., the variables are always consistent with the constraints. The ability to model geometric relations is important for the alignment of real and virtual objects. It is also important for clear visualization. The ability to model human-computer interface is important for the communication of the user with the system and his coworkers. The ability to model space and time dependencies is important for multimedia and for presentation applications such as tutoring systems.

4 Architecture proposal

For easy reference, an architecture model is proposed here. In this model, we suggest supporting a number of vision algorithms, implementing a constraint system base for declaring and processing the application constraints, an extension system based in agents and modules for output and multimedia communication.

4.1 Vision algorithms

Vision algorithms can be responsible for most input data. Low level vision algorithms can rely on the constraint-based system for handling high level problems such as occlusion. Usual vision algorithms for our context are object recognition, pose estimation, tracking, surface reconstruction, motion analysis and matching between views.

4.2 Constraint-based application definition

The core of the application definition can be declared as a combination of constraints and agents. First-order constraints are only able to model reactive behavior. They are useful to solve those typical problems based on constraint satisfaction, but they are also useful to model real-world objects and passive virtual objects for direct manipulation. Instead of providing support for higher-order constraints (conditionals) or side effects (procedure calls), we choose modeling the active portion of the application as agents. The constraint system specifies the behavior of the environment where the agents inhabit. Dix *et al.* [94] propose the Agents, Medium and Objects (AMO) model for interface analysis and suggest implementing the interface as the medium. We propose a constraint system modeling real and virtual object behavior as the medium.

4.3 Agent-based application specific extensions

While the constraint system models the reactive behavior of the environment, agents can model intelligent autonomous behavior. Agents are able to access some variables and constraints of the constraint system for reading or writing, corresponding to the sensorial and motor functions. The architecture for agent implementation is not relevant so long as it runs in parallel with the constraint engine because only the agent behavior is relevant. Agents can not only model active components of the application but also users, robots and previously unknown components. We believe this framework can help to establish a formal model of user activity in cooperative work.

4.4 Actuators, rendering, multimedia and telepresence

Signals for robotic devices, force-feedback output and displays are generated by processing the output variables of the constraint system. A haptic rendering system based on constraints can be found in [Zilles and Salisbury, 95]. The constraint system can also participate in rendering control. In section 5.3, we describe a rendering control strategy for displaying labels. Multimedia orchestration can be represented by the constraint-based model, one example of constrained based multimedia document model is Madeus [Jourdan *et al.*, 98]. Constraint systems are an alternative to scripting in presentations. Telepresence can be emulated using virtual objects such as avatars.

5 Results in application scenarios

We analyze some application scenarios where the adequacy of constraint-based systems for our purpose is quite evident. Each scenario is meant to handle a different aspect of constraint programming and the applicability to AR and CSCW. Experiments for the scenarios 5.1 (virtual theatre) and 5.3 (annotation) were implemented and results are shown. For the remaining scenarios, we made reference to related work and pointed how to use constraints in each case.

5.1 Virtual theatre, MUD

Virtual theatre can be used for entertainment, art and education. There are collaborative systems for role-playing named MUD (multiple user dialogue), most of them is text-based and desktop-based. Collaborative virtual theatre is, therefore, an improvement of MUD. Inspired by puppet theatre, we present an AR application scenario where a real marionette is able to play the role of a virtual creature. This can be used for collaborative storytelling, where each child controls virtual characters like marionettes or body parts of them in cooperation.

We implemented an off-line prototype AR system for the animation of a virtual marionette. A movie of a real marionette was recorded. The control rod of the marionette was prepared with four landmarks (enough for pose estimation). The camera was previously calibrated. A CSG model of the marionette was created with some free parameters, as shown in *figure 1*. A set of constraints (2) and (3) relating these parameters to the position of the landmarks in 3D space was constructed. We chose a gradient descent optimization framework for partial constraint fulfillment. The virtual marionette is rendered with

POV-Ray™ [POVRAY] and the resulting animation is the composition of the original video and the virtual marionette. Results can be appreciated in *figure 2*. More frames are presented in *figure 6*.

```
distance(b,n3)=d1;                    (2)
distance(h,n1)=d2;
distance(n1,n2)=d3;
distance(n2,n3)=d4;
distance(n1,l1)=d5;
distance(l1,f1)=d6;
distance(n3,l2)=d7;
distance(l2,f2)=d8;

distance(a6,h)<d9;
distance(a1,f1)<d10;
distance(a2,f2)<d11;
distance(a5,b)<d12;

f1.y>0; f2.y>0;
n1.y=h.y;
n3.y=b.y

less strict:                          (3)

f1.y=0; f2.y=0;
h.y=0; b.y=0;
n2.y=0;
```

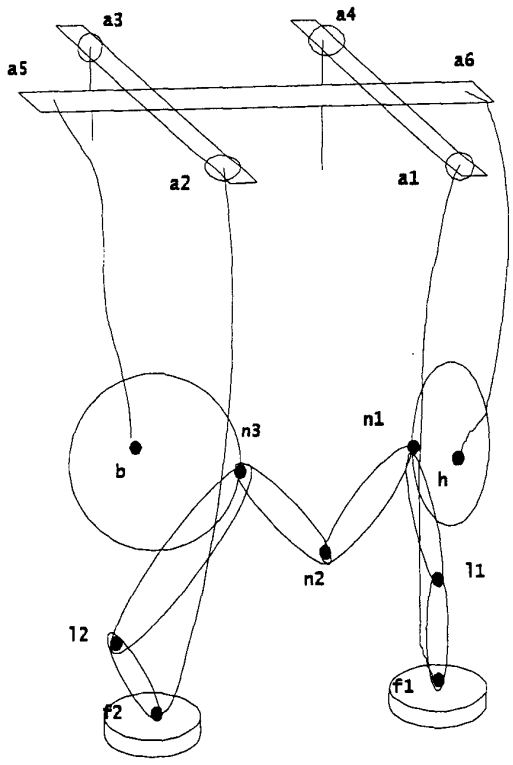
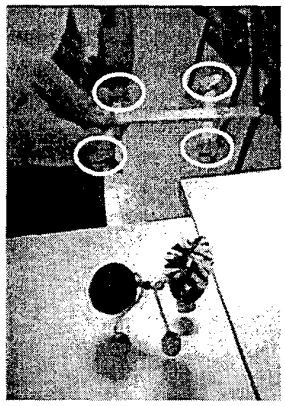
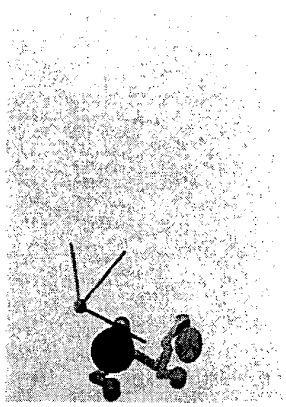


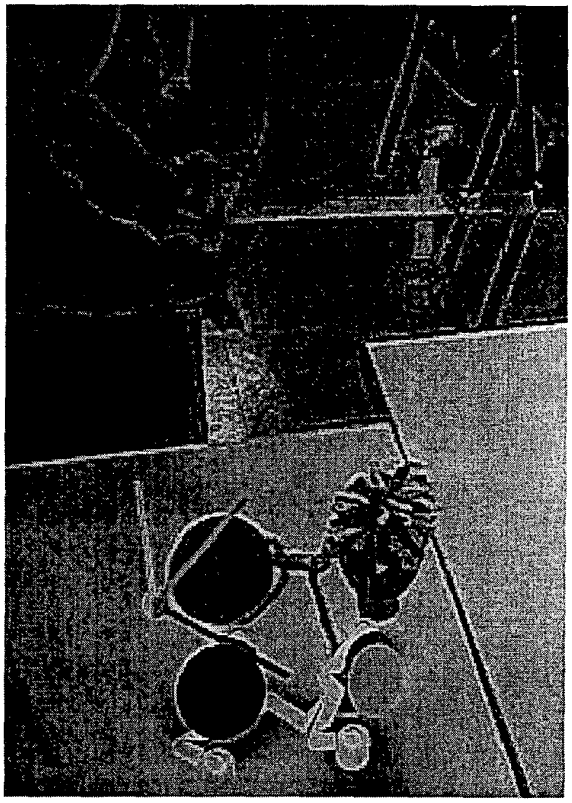
Figure 1 – Virtual marionettes. Constraints are defined to model the skeleton, the control rod and the relation between them.



(a)



(b)



(c)

Figure 2 – Virtual marionettes. (a) frame #19 from original movie, landmarks are highlighted, (b) image of virtual model for frame #19, (c) composition of virtual object and real world, frames #19 from the augmented movie.

The constraints (**2**) used in the modeled animated body are Euclidean distance between points and positive y-coordinate for all points, where negative values mean beneath the floor level. In order to obtain the gravity effect, minimization of y-coordinate for each point is represented as less strict constraints (**3**) for partial constraint satisfaction.

Pose estimation was based in the detection of three of the landmarks and the distance between each pair of landmarks. Correspondence between an image point and the associated landmark is maintained through the image sequence based on the closest solution to the result from the previous frame.

The role of the constraint-based system in this case can be seen from many perspectives: the system is used to align virtual objects to real ones; the system collects 3D coordinates as input (like optical motion capture) and uses them as control points for an animation; the system models the behavior of a real object and tries to mimic it when receiving similar input.

5.2 Augmented tutoring systems

Tutoring systems should be based on interactive presentations. Interface agents should be employed to simulate a human instructor. The agent must be able to convince the user that it can actuate in the real environment. A constraint-based system can model the real and virtual parts of the environment. The agent should sense the environment through accessible variables of the constraint system and actuate changing those write-enabled variables. Sequence of events can be modeled by "before", "after" and "during" time relations, becoming a scripting system. For an example of an agent-based virtual tutoring system, see Steve [Rickel and Johnson, 98].

The roles of constraints in this case are modeling time dependencies, modeling a working real system where people can manipulate in both the real and the virtual case and serving as a protocol for communication among participants of the collaborative systems, i.e., agents.

5.3 Annotation

The problem of displaying labels well positioned for readability and appropriate spatial semantics can also be modeled in the constraint framework. We implemented a readable annotation system, using constraints to avoid label overlapping and to keep each label close to the corresponding object. The object position is detected using image processing. The objects were rendered in POV-Ray™. The employed constraints are shown in **(4)**. The geometric scheme can be seen in *figure 3*.

```
non_overlap(c1,c2);                    (4)
non_overlap(c1,c3);
non_overlap(c2,c3)

less strict:
distance(a1,c1)=0;
distance(a2,c2)=0;
distance(a3,c3)=0;
```

The role of constraints here is modeling spatial dependencies for appropriate visualization. The constraints were designed with readability of the information and spatial relation semantics in mind. See results in *figure 4*.

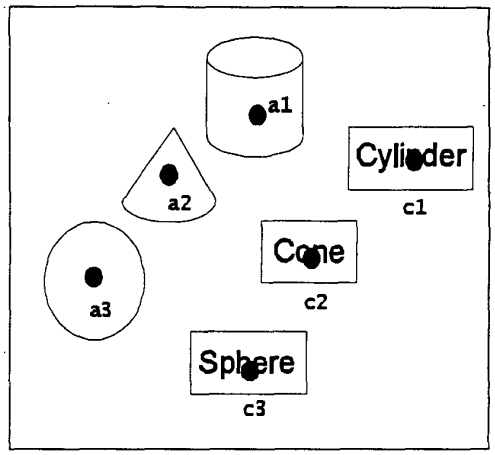


Figure 3 – Annotation system. Variables are rectangular label centroids and anchors in real world objects.

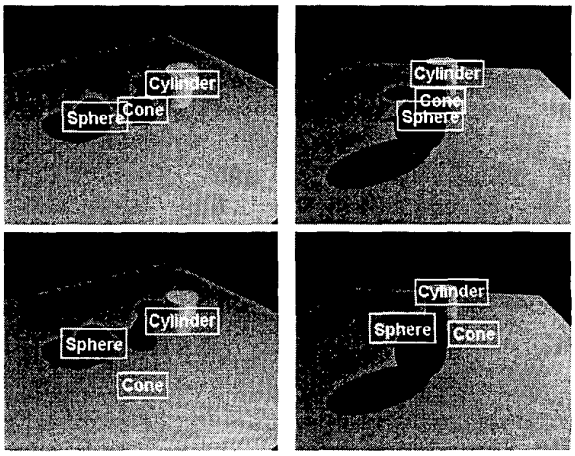


Figure 4 – Annotation system. Upper: labels are placed right on the anchored positions, lower: a constraint system corrects the overlap and semantic problem.

5.4 Design and manufacture

For tasks such as design and manufacture in the real world, the system can provide users with real and virtual tools as interface metaphors. One example of metaphor is the ruler metaphor for the object alignment problem [Raisamo and Raiha, 96]. In the desktop case, a mouse-controlled ruler pushes objects with the purpose of aligning them. Extending the desktop idea to AR, a real world ruler can be used to push real and virtual objects. The alignment can be implemented as a set of constraints activated when the ruler is on the workspace. Another interface technique that can be imported from the desktop to the augmented workspace is snap-dragging [Gleicher and Witkin, 91]. In the desktop, snap-dragging holds the object being dragged in valid positions. The user can only move the object in its constrained domain. For AR systems, snap-dragging can be implemented using real objects as control

instruments for virtual objects. In *figure 5*, the (hypothetical) system should provide guidance for alignment of two real boxes. Using the augmented version of snap-dragging, the virtual contour of the box shows the closer position where the real box will satisfy the specified alignment constraints. Differential manipulation [Gleicher and Witkin, 91] is a technique for incremental update of constraints face to bounded continuous change of input. This technique is also very useful in the AR context.

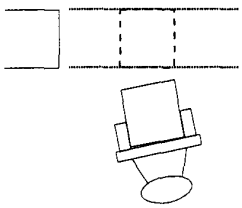


Figure 5 – Snap-dragging.

In this case, constraints are used to implement interface metaphors importing spatial dependencies from real-world situation and using an artificial spatial dependency (snap-dragging) imported from graphics software.

6 Conclusion

We discussed in this paper the potential of employing constraint-based systems for the specification of augmented reality applications. We showed that the combination of these technologies is synergistic. Our greatest motivation was the creation of workspaces away from the desktop where workers can collaborate using real and virtual tools. An architecture model mixing constraints, agents and augmented reality was proposed. Typical examples and corresponding results were presented.

Acknowledgements

The first author (C. H. Q. F.) was supported by CNPq (Proc. 140800/00-0).

References

1. Azuma, R. T., 1997, A Survey of Augmented Reality. In *Presence: Teleoperators an Virtual Environments 6*, 4, pp 355-385.
2. Badros, G. J.,1998, Constraints in Interactive Graphics Applications. http://citeseer.nj.nec.com/badros98constraints.html
3. Badros, G. J., Nichols, W. P. J., Borning, A., 2000, *A Constraint Extension to Scalable Vector Graphics*. University of Washington Technical Report 200-08-04.
4. Baker, M., Bricker, L, Fujioka, E. and Tanimoto, S., 1998, *Colt: A System for Developing Software that Supports Synchronous Collaborative Activities*. University of Washington Technical Report UW-CSE-98-09-03.
5. Barták, 1999, R., Constraint Programming: In Pursuit of the Holy Grail. http://citeseer.nj.nec.com/276330.html
6. Benhamou, F., 1995 Interval constraint logic programming. In A. Podelski, editor, *Constraint Programming: Basics and Trends*. LNCS 910, March 1995.
7. Buford, J. F. K., 1994, *Multimedia Systems*. ACM Press.
8. Chenney, S., 1993, Sced: Constraint Based Scene Editing. http://citeseer.nj.nec.com/367518.html
9. Deligiannidis, L., 2000, *DLoVe – A specification paradigm for designing distributed VR applications for single or multiple users*, http://citeseer.nj.nec.com/deligiannidis00dlove.html
10. Diehl, S., Keller, J., 2000, *VRML with constraints*. In Proceedings of the Web3D-VRML fth symposium on Virtual reality modeling language, Monterey, California. http://www.cs.uni-sb.de/RW/users/diehl/VRMLCONSTR/VRMLConstr.html
11. Dix, A., Finlay, J., Hassell, J., 1994, Environment for Cooperating Agents: Designing the Interface as a Medium, In *CSCW and Artificial Intelligence*, J. H. Connolly and E. A. Edmonds (Eds.), Springer-Verlag.
12. Fluckiger, F., 1995, *Understanding Networked Multimedia*, Prentice Hall.
13. Freeman-Benson B.N., Maloney J., Borning A., 1990, An Incremental Constraint Solver. In *Communications of the ACM*, 33, pp. 54-63.
14. Frünwirth, T., Herold A., Küchenhoff V., LeProvost T., Lim P., Monfroy E., Wallace M., 1992, Constraint Logic Programming – An Informal Introduction, *Lecture Notes in Artificial Intelligence*, vol 636, pp3-35.
15. Gleicher, M., Witkin, A., 1991, Differential Manipulation, *Proceeding of Graphics Interface '91*, pp 61-67.
16. Gleicher, M., 1993, A Graphics Toolkit Based on Differential Constraints, *ACM Symposium on User interface Software and Technology*, pp 109-120.
17. Graf, W. H., 1996, The constraint-based layout framework LayLab and its applications, *Proceedings of the DKFI Workshop on Constraint-Based problem Solving*, DKFI GmbH, Kaiserslautern.
18. Helm, R., Huynh, T., Marriot, K., Vlissides, J., 1995 An Object-Oriented Architecture for Constraint-Based Graphical Editing. In C.Lara, E. H. Blake, V. de Mey, X. Pintado (eds.), *Object-Oriented programming for Graphics, Focus on Computer Graphics, Tutorials and Perspectives in Computer Graphics*, Springer-Verlag, pp 217-238.
19. Hower, W., Graf, W. H., 1996, A bibliographical survey of constraint-based approaches to CAD, graphics, layout, visualization and related topics. *Knowledge Based Systems*, 9(7), pp 449-464.
20. Jourdan, M., Layaida, N., Roisin,C., Sabry-Ismail, L., Tardif,L., 1998, Madeus, an Authoring Environment for Interactive Multimedia Documents. In *Proceedings: ACM Multimedia '98*, Bristol UK.
21. Kramer, G. A., 1992, A geometric constraint engine. *Artificial Intelligence* 58, pp 327-360. Also in *Constraint-Based Reasoning*, E. C. Freuder, AA. K. Mackworth (eds.), MIT/Elsevier pp 327-360.
22. Kumar, V., 1992, Algorithms for constraint-satisfaction problems: A survey. *AI Magazine*, 13(1), pp 32-44.
23. Kurlander, D., 1993, *Graphical Editing by Example*. Ph.D. Thesis. Columbia University. Computer Science.
24. Leler, Wm., 1988, *Constraint Programming languages: Their Specification and Generation*, Addison-Wesley.
25. Myers, B. A., 1996, Easily Adding Animations to Interfaces Using Constraints, in *Proceedings UIST'96: ACM SIGGRAPH Symposium on User Interface Software and Technology*, Seattle, WA.
26. Montanari, U. and Rossi, F., 1995, Concurrency and Concurrent Constraint Programming. In A. Podelski, editor, *Constraint Programming: Basics and Trends*. LNCS 910.
27. Olsen Jr., D. R., 1992, *User Interface Management Systems: Models and Algorithms*, Morgan-Kauffmann.
28. POV-Ray™ homepage, http://www.povray.org
29. Raisamo, R., Raiha, K. J., 1996, A New Direct Manipulation Technique for Aligning Objects in Drawing Programs, *Proceedings of The Symposium on User Interface Software and Technology*, Seattle, Washington, pp 6-8.
30. Rickel, J., and Johnson, W. L., 1998. Animated agents for procedural training in virtual reality: Perception, cognition, and motor control. *Applied Artificial Intelligence*.
31. Rossi, F., 1999, *Constraint Logic Programming*, http://citeseer.nj.nec.com/422446.html
32. Ryall, K., Marks, J., Shieber, S., 1996, An interactive system for drawing graphs, In Stephen North, editor, *Graph Drawing*, LNCS 1190, pp 387-393. Springer-Verlag. *Proceedings of the Symposium on Graph Drawing*, GD '96.
33. Schmalstieg, D., Fuhrmann, A., Hesina, D., Szalavari, Zs., Encarnação, L.M., Gervautz, M. and Purgathofer, W. 2000, The Studierstube Augmented Reality Project. Submitted for publication. Available as technical report TR-186-2-00-22, Vienna University of Technology.
34. Wallace, M., 1995, *Survey: Practical applications of constraint programming*, Technical Report, IC Parc. Also in *Constraints Journal*, 1(1).
35. Winston, P. H., 1992, *Artificial Intelligence*, 3rd edition, Addison-Wesley.
36. Witkin, A., Fleischer, K., Barr, A., 1987, Energy Constraints on Parameterized models, *Computer Graphics*, vol.21, pp.225-232.
37. Witkin A., Kass, M., 1988, Spacetime constraints. In John Dill, editor, *Computer Graphics* (SIGGRAPH '88 Proceedings), volume 22, pp 159-168.
38. Zilles, C.B., and Salisbury, J.K., 1995, A Constraint Based God-Object Method For Haptic Display, in Proc. *IEE/RSJ International Conference on Intelligent Robots and Systems, Human Robot Interaction, and Cooperative Robots*, Vol 3, pp. 146-151.

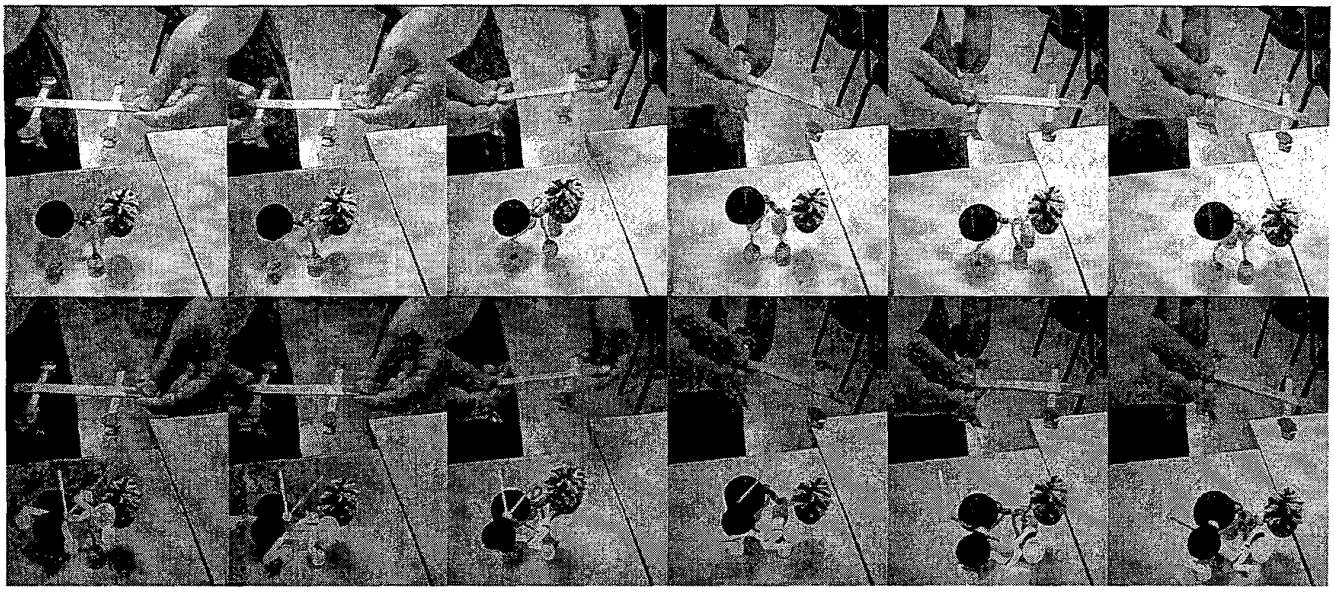


Figure 6 – Virtual Marionettes. More frames from the original and the augmented videos.