(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
19 April 2001 (19.04.2001)

PCT

(10) International Publication Number
WO 01/27835 A1

(51) International Patent Classification⁷: G06F 17/60

(21) International Application Number: PCT/US00/27849

(22) International Filing Date: 6 October 2000 (06.10.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/158,503	8 October 1999 (08.10.1999)	US
Not furnished	6 October 2000 (06.10.2000)	US

(71) Applicant: AUTONOMOUS EFFECTS, INC. [US/US]; Suite 208, 10 Avocet Drive, Redwood City, CA 94065 (US).

(72) Inventor: ANDERSSSON, Russell; 2 Carrie Lane, Malvern, PA 19355 (US).

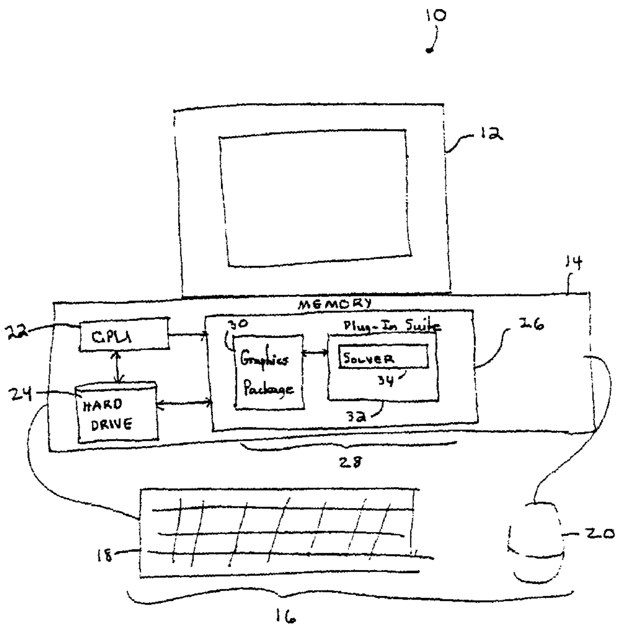
(74) Agent: LOHSE, Timothy, W.; Gray Cary Ware & Freidenrich LLP, 400 Hamilton Avenue, Palo Alto, CA 94301-1825 (US).

(81) Designated States *(national)*: AU, JP, NZ.

(84) Designated States *(regional)*: European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published:
— *With international search report.*

(54) Title: SYSTEM AND METHOD FOR EFFICIENTLY SOLVING COMPLEX SYSTEMS OF TIME-VARYING AND TIME-INVARIANT VARIABLES



(57) **Abstract:** A system and method for solving complex systems having time-varying and time-invariant. The system comprises a software application (32) which may include a graphics package (30) and a suite of plug-in programs (32) that enhance the operation of the graphic package. The software application (32) may also include a solver (34) in accordance with the invention which solves the complex systems generated by the computer graphics package.

SYSTEM AND METHOD FOR EFFICIENTLY SOLVING COMPLEX SYSTEMS OF TIME-VARYING AND TIME-INVARIANT VARIABLES

Background of the Invention

This invention relates generally to a system and method for solving for the values of unknown variables, subject to desired objectives and relationships, wherein some of the variable values may change over time and others cannot. This method may be used to solve complex systems. In a preferred embodiment, the system and method may be applied to computer graphics where it can be applied to force complex animated mechanisms to satisfy certain constraints, or to match three dimensional, animated elements to corresponding elements in a video or film clip.

In the past, there have been software applications which are designed to solve a particular problem within a particular application. Thus, there are software applications which are designed to specifically solve computer graphics equations to derive the output for the computer display. There are other software applications which solve problems for other complex systems. These conventional software applications may use similar strategies for solving the complex system such as using various typical search algorithms. However, none of the conventional software applications are general enough to be used to solve a variety of different complex systems. Thus, it is desirable to provide a system and method for efficiently solving complex systems of time-varying and time-invariant variables which may be used for a variety of different applications and it is to this end that the present invention is directed.

Summary of the Invention

A system and method for solving complex system is described wherein the solution of the complex system comprises values of unknown variables, subject to desired objectives and relationships, and further where some of the variable values in the complex system may change over time and others cannot. The system and method efficiently solves these complex systems without limitation to the dependencies among the variables. This method has application generally to solving various complex systems. In a preferred embodiment as described below, the solving method may be used for computer graphics where it can be applied, for example, to force complex animated mechanisms to satisfy certain constraints or to match three dimensional, animated elements to corresponding two dimensional elements in a video or film clip.

Thus, in accordance with the invention, a system and method for solving complex systems of time-varying and time-invariant variables is provided. The method comprises explicitly identifying dependencies between variables, analyzing the variable dependencies to identify groups of variables that are mutually interdependent ("loops"), ordering the loops, based on inter-loop dependencies, from most independent to most dependent, and for any time of interest, applying a solving step to each loop in turn, following the dependency ordering established above. A system is also described.

The method may further comprise clearing any previously computed solutions if anything on which those solutions depend is changed. The steps of loop identification and loop ordering may be accomplished simultaneously. In accordance with the invention, any variable or objective can be deactivated (excluded from consideration in the solution) for specified time(s). In accordance with the invention, any variable can be constrained to be invariant over its active times, and the solving step may further comprise dividing the loop variables into two lists, one containing time-invariant variables, and the other containing time-variant variables, identifying the range of times that encloses the active times of the time-invariant variables ("invariant range"), recursively ensuring that all prior loops that this loop depends on have been solved for all times in the invariant range, and, for the time of interest, applying a solving step that simultaneously computes both the time-varying and the time-invariant variable values.

The step of dividing the loop variables into two lists and identifying the invariant range are accomplished once, possibly simultaneously with loop identification and/or loop ordering, since this analysis is valid for solution at any time. In solving the loops, a standard multivariate optimization method that systematically explores potential variable values to optimize an objective function may be used.

Brief Description of the Drawings

Figure 1 is a diagram illustrating a computer system that may include the solver system and method in accordance with the invention;

Figure 2 is a flowchart illustrating a preferred embodiment of a method for solving complex computer graphics in accordance with the invention;

Figure 3 is a flowchart illustrating more details of the analysis steps of the method of Figure 2;

Figure 4 is a flowchart illustrating more details of the solving steps of the method of Figure 2;

Figure 5 is an example of simple problem that is solved using the solving system in accordance with the invention; and

Figure 6 is an example of a more complex problem that is solved using the solving system in accordance with the invention.

Detailed Description of a Preferred Embodiment

The invention is particularly applicable to solving computer graphics complex systems and it is in this context that the invention will be described. It will be appreciated, however, that the system and method in accordance with the invention has greater utility, such as to solving equations for other types of complex systems, such as machine vision systems, user interface systems, and robotic systems. Prior to describing a preferred embodiment of the invention, various definitions of terms that will be used in this application will be described. These definitions may aid in the understanding of the invention as described.

Definitions

1. Actuator: An actuator is a component which outputs a designated value or values. The actuator may be used in computer graphics applications to specify and control aspects of a 2-D or 3-D scene, such as an object's position, orientation, size, color, etc. Actuators effectively represent the variables to be solved for wherein the actuators may be normal actuators or steady actuators. A normal actuator has a value that changes from frame to frame of the computer graphics whereas the steady actuators have values that must be held constant throughout an entire animation. In other words, steady actuators produce the same value for all time values (for an entire animation) while normal actuators produces a potentially different value for each time (or each frame of the animation). The described method produces values to be output by each actuator in the scene to meet designated objectives which are expressed as comparators (described below). Each actuator references one or more comparators as described below.

2. Steady Actuators: The steady actuators produce only a single value(s) for every time in the animation sequence. Steady actuators have a user-defined time range over which their comparators are evaluated in the process of determining the optimal value.

3. Comparator: A comparator is a component expressing a desired objective and/or relationship. Those desired relationship of objective may include for example, a desired relationships among elements in a 3-D scene, such as that two

objects be located at the same location in the 3-D scene or a 2-D image view, or that several objects be co-linear. Each comparator produces one or more measurements, which ideally are zero indicating that the desired relationship/objective has been obtained.

4. Dependency: A first variable or other object depends on a second variable or object if the second variable's value must be known to compute the first variable's value. For example, in the equation $x = (y + 3)/z$, x depends on y and z.

5. Loop: A loop is a collection of interrelated mutually dependent scene objects or variables. For example, A's value depends on B, B's value depends on C, and C's value depends on A, where one or more of A – C are actuators. Loops can become substantially more complex and a scene may contain any number of loops. For example, a loop, L1, might depend on another loop, L2, but if L2 also depends on L1, then L1 and L2 are not disjoint but are in fact a single loop.

6. Time of Interest: A loop must generally be solved only for a particular time (the "time of interest") to update the software's display windows. The user may later request a solution for a different time of interest and we will attempt to do so with a minimal expenditure of computer time.

7. Cache: A cache is a data storage area consisting of pairs of data (time, value) wherein each pair defines an actuator's output at a particular time as specified by the data. The cache may contain no entries (after it is flushed), or any number of

entries for unique times. If the cache contains an entry for a particular time, it is said to be valid for that time. If the cache does not contain an entry for a particular time, the cache is invalid at that time. A cache may be implemented by any means such as linked lists, bit arrays, and so on as may be apparent to those skilled in the art.

Description

Now, a preferred embodiment of the invention will be described in which a complex system having time-varying and time-invariant actuators may be solved using the method. In particular, the invention in the preferred embodiment may be applied to solve a complex computer graphics system as will now be described.

Figure 1 is a diagram illustrating an example of a computer system 10 that may include the solver system and method in accordance with the invention. The computer system 10 may include a display 12 (any typical display such as a CRT or a LCD), a processing unit 14 connected to the display 12 and one or more input/output peripheral devices 16 connected to the processing unit such as a keyboard 18 and a mouse 20. The display permits images to be displayed to the user of the computer system while the input/output devices permit the user to interact with the software application being executed by the computer system. For example, a computer graphics software application may display an animation on the display and the animation may be modified by the user inputting commands using the keyboard or mouse.

The processing unit 14 may further include a central processing unit (CPU) 22, a persistent storage device 24 and a memory 26. The persistent storage device, which may be a hard disk drive, a tape drive, an optical drive or the like, may store one or more software applications and an operating system that may be executed by the computer system. To execute a software application, it is typically downloaded from the persistent storage device into the memory and then the CPU 22 executes the software application. The CPU may also control the operation of the computer system. In the computer system shown in Figure 1, one or more software applications 28 are shown already resident in the memory so that they may be executed by the CPU.

In this example, the one or more software application may include a graphics package 30 and a suite of plug-in programs 32 that enhance the operation of the graphics package. As described in more detail in a user manual which is attached to this application as an appendix and is incorporated herein, in a preferred embodiment, the graphics package may be Studio 3D Max , a commercially available graphics package, and the suite of plug-ins may be referred to as the SceneGenie software application. The SceneGenie software application 32 may include a solver 34 in accordance with the invention which solves the complex systems generated by the computer graphics package. Examples of the solution provided by the solver 34 in accordance with the invention are provided with respect to Figures 5 and 6. Now, a preferred embodiment of a method for solving complex systems in accordance with the invention will be described.

Figure 2 is a flowchart illustrating a preferred embodiment of a method 40 for solving complex computer graphics in accordance with the invention. The disclosed method shows how to determine values for the actuators of the graphics program when the graphics scene consists of a mixture of steady and normal actuator types as defined above. The presence of steady actuators makes this solving process substantially more complex, but offers the user substantial benefits as more information can be extracted from a scene. In step 42, the computer graphics scene is analyzed to identify the normal and steady actuators as will be described below in more detail with reference to Figure 3. The output of the analysis step is a list of loops in the scene and, for each loop in the scene, a list of the steady and normal actuators. In step 44, the method solves the steady and normal actuators for each loop for the steady actuator values and the normal actuator values. This step is described in more detail below with reference to Figure 4. Once the final steady and normal actuator values have been determined by the solving step, those determined values may be applied to the computer graphics scene to change the scene based on the changed steady and normal actuator values. Now, more details of the analysis step in accordance with the invention will be described.

Figure 3 is a flowchart 50 illustrating more details of the analysis steps of the method of Figure 2. In particular, the analysis steps may include one or more steps in which the scene is analyzed to extract the necessary information to determine the current values of the steady and normal actuators. In step 52, the method begins by analyzing the scene to locate all of the loops in the scene. For any element of the scene, the method posits that there exists a way to determine the other elements which the element depends on. In particular, these dependencies are often listed explicitly, but others may be deduced from the type of the element in an application-dependent manner. Each actuator in the scene is located and its dependencies are enumerated, starting from its comparators and proceeding to examine everything its comparators depend on, everything they depend on, and so on recursively, until there are no remaining unexamined dependencies. During this process, the actuator will generally be found to depend on other actuators or at least depend upon itself. The list of dependent actuators for each actuator constitute the raw loop information.

In step 54, the generated raw loop information may then be further examined to consolidate any loops and order the loops. In particular, during the consolidation process, each loop is repetitively compared to the other loops. If the two loops are mutually interdependent (i.e., the dependents of one loop contains members of the other loop and the dependents of the other loop contain members of the first loop), then the two loops are not disjoint and are consolidated into a single loop. Otherwise, the two loops are left as two separate loops which will be solved separately.

During the consolidation process, it becomes apparent when one loop depends on another loop, but not the reverse. In this case, a reordering process may occur in which the two loops are reordered so that the more independent loop immediately precedes the dependent loop in the loop list. Thus, as the loop consolidation process proceeds, the loops are simultaneously sorted such that non-dependent loops come early in the list, and the more dependent loops appear later in the list. The sorting (ordering) and consolidation process continues until the loops are all disjoint, and until no loop depends on a loop which comes after it in the list.

Once this has been done, one can see that the loop-solving process must start at the beginning of the list of loops, solving each loop, and working through to the last loop in step 56. During this process, the more independent loops are processed earlier since they appear earlier in the list. Then, the more dependent loops, which depend on the more independent loops already processed, may be processed efficiently since the data from the independent loops is already available. Thus, the necessary data values from the more independent loops on which the currently processed loops depends is already available which makes the processing of the loops must faster.

Now, the processing of each loop is described. In particular, each loop may consist of a mixture of steady and normal actuators. The loops must be solved for only the single time of interest corresponding to the effective time the user is being shown the display window, but the process can be iterated as required. Thus, the remainder of the steps described below are repeated, in order, for each loop in the loop list.

In step 58, the method checks whether the loop does in fact need to be solved. In particular, each actuator maintains a cache consisting of times and values at which its optimal value has already been determined. Steady actuators are valid for either all or no times, with a single optimal value. It may be convenient to consider this optimal value to correspond to time zero for book-keeping purposes. In operation, the actuator caches are flushed (emptied of the values and times) when objects they depend on are changed, typically by user action. Often a user action will affect only a small number of loops, and we wish to avoid solving still-valid loops to save computer time. Thus, the method checks whether a loop must be solved by checking the caches of each actuator that is a member of the loop. If each actuator of the loop is still valid at the time of interest, then the loop can be skipped with no further work and the method loops back to step 56 to process the next loop. If the loop needs to be processed (i.e., some of the caches are invalid for one of the actuators in the loop), then the method processes the current loop.

In step 60, the actuators of the current loop are sorted into two lists wherein one list consists of steady actuators and the other list consists of the normal actuators. During this sorting process, it is convenient to determine a single loop time range which encloses the range of all the steady actuators in the loop. In step 62, the method determines if there are any steady actuators in the current loop. If there are no steady actuators in the current loop, the method skips to step 70, solves the normal actuators and loops back to step 56 to process the next loop.

If there are steady actuators in the loop currently being processed, the method ensures, in step 64, that each loop that this loop depends on has been solved for at all frames in this loop's range (i.e., check to ensure that all precedent conditions have been solved). Since only solutions at the time of interest are required as the method's ultimate output, the earlier loops need only be guaranteed to have been solved at the time of interest. Solving for the current loop's values requires solutions for the prior loops for all times within the present loop's range. To achieve this, the method examines the loops preceding the current loop in the loop list to see if the preceding loop's members include any of this loop's dependencies. Any steady actuators will already have been solved for, by construction. For the normal actuators in the earlier loop, the method iterates over each frame time and checks whether each actuator's cache is valid at that time. If not, a solve cycle must be initiated for that loop at that time, as described below with reference to Figure 4. At the conclusion of this solving process, it should be clear that at any time within the loop range, any actuator or object within the present loop can be evaluated, and all other information required will already be cached and available. This is crucial to successful and efficient implementation. The loop may now be solved in step 66 for its steady actuators, as described with reference to Figure 4, with the special objective function as described below.

In step 68, the method determines if the normal actuators must be solved at the time of interest by examining the validity of their caches. This may occur if the loop did not contain any steady actuators (as described above) or if the range of the steady actuators did not contain the time of interest. If the normal actuators are not being solved, then the method loops back to step 56 to process the next loop in the list. Otherwise, in step 70, the normal actuators are solved using the solving step described below with reference to Figure 4.

The steady and normal actuators in a loop are then solved in an efficient manner since the loops are arranged and processed in dependency order. To optimize the interactive performance of this method, it is possible to allow the user to specify that the comparators of the steady actuators are evaluated only at spaced out intervals so that the comparators are not evaluated as frequently. For example, the comparators may be evaluated every N frames. As an example, this parameter may be set at 5 so that only 1/5th of the number of frames must be solved for in step 66 and in evaluating the objective function below, affording a significant speedup in processing time. Now, the solving step in accordance with the invention will be described.

Figure 4 is a flowchart illustrating more details of the solving steps 80 of the method of Figure 2. In particular, the solving steps are applied to both the steady actuators and the normal actuators. In step 82, a list of actuators to be solved for is received which are the input to the solving steps. The list received is either all steady actuators or all normal actuators depending on whether steady actuator values or normal actuator values are being solved for. In step 84, a solving technique is applied to the list of actuators. In a preferred embodiment, the solving step implements a standard multivariate optimization method, such as was described in A Variable Metric Method (Numerical Recipes in C++: The Art of Scientific Computing, W.H. Press, S.A.Teukolsky, W.T. Vetterling, B.P. Flannery, Cambridge University Press, 2nd Ed., 1992). In using this optimization method, the actuators constitute the variables and the optimization method systematically explores their potential values to optimize an objective function as described below. The optimization method returns tentative values to the actuators which produce those values when the objective function is evaluated. Once the optimal actuator values have been determined, they are entered into the actuators' caches, and the solving procedure terminates. Now, the objective function in accordance with the invention may be determined as follows.

In the preferred embodiment, the objective function for normal actuators consists of the sum of squares of the measurements of each comparator referenced by the actuators. The comparators are typically referenced by more than one actuator in a loop and these duplications must be detected so that each measurement counts only a single time in the final objective function, to avoid inappropriate biasing. In the preferred embodiment, the objective function for steady actuators consists of the sum of the squares of the measurements, accumulated over each time in each actuator's range. The objective function evaluator iterates over each time in the range, and at each time, it executes an entire Solve cycle for the normal actuators that are part of this loop, with the steady actuators held at their tentative values. By reusing earlier solutions as seed values, the normal-mode solve cycle generally converges rapidly.

With the tentative steady actuator values and the tentative-optimum normal actuator values in place, each steady-actuator comparator is then evaluated, and the results (squared) added up without duplication. Some comparators may be active during only a portion of the range; inactive comparators may be skipped at this point. In the preferred embodiment, the same computer code can handle both normal and steady actuator solving, using C++ language features to redefine the objective function for each case. In a preferred embodiment, separate solver instances (of the same C++ class) may handle the steady-mode and normal-mode solving cycles since they happen simultaneously. Now, two examples of problems that may be solved using the solver system and method in accordance with the invention will be described.

Figure 5 is an example of simple problem that is solved using the solving system in accordance with the invention. In particular, a rod, R, is shown that rotates in three dimensions (3D) about a point, B, and the goal it to solve for the 3D rotation of the rod that brings a point, A, as close as possible, in three dimensions, to a target point, P. It is possible that point P may be moved or may be animated to follow a path. For this problem, the variables are the 3D rotation of the rod and the goal is to minimize the distance between points A and P. In this simple example, there is only one loop which is the rotation of R and the positions of P and A that are interdependent. Using the solver in accordance with the invention, a solution is found wherein, for any given time of interest, t, and therefore any known position of P, the rotation of R is determine that minimizes the distance between P and A. Now, another example of a problem solved using the invention will be described.

Figure 6 is an example of a more complex problem that is solved using the solving system in accordance with the invention. In this example, it is desirable to solve for the 3D position and rotation of a head, H, that matches the appearance of the head in image, I. Specifically, the 3D position and rotation of the head H should match the 2D position of the 2D features, such as a first feature, F_1, in a source image I with the 2D position that the corresponding features, such as F_2, would occupy in an image created when H is viewed through a camera, C. Since camera C may move, its position or rotation must also be determined based on a background feature, B. In this more complex example (which is still simpler than more complex examples that may be solved with the solver in accordance with the invention), note the interrelationships between the camera and the head. In this example, the position of the head and camera and time-varying. If the camera was stationary, its position and rotation must still be determined, but those variables are time-invariant.

While the foregoing has been with reference to a particular embodiment of the invention, it will be appreciated by those skilled in the art that changes in this embodiment may be made without departing from the principles and spirit of the invention as defined by the appended claims.

CLAIMS:

1. A method for solving complex systems of time-varying and time-invariant variables to optimize the satisfaction of goals, comprising:

explicitly identifying dependencies between variables;

analyzing the variable dependencies to identify groups of variables that are mutually interdependent to identify one or more loops;

ordering the loops, based on inter-loop dependencies, from most independent to most dependent; and

for any time of interest, applying a solving step to each loop in turn, following the dependency ordering established above.

2. The method of claim 1 further comprising clearing previously computed solutions if anything on which those solutions depend is changed.

3. The method of claim 1, wherein loop identification and loop ordering are accomplished simultaneously.

4. The method of claim 1 further comprising excluding from solution one or more of a variable and a goal for specified time(s).

5. The method of claim 4 wherein each goal is specified as a computation ("comparator") yielding one or more measurements, which are optimally zero.

6. The method of claim 5 further comprising constraining one or more variables to be invariant over its active times, and wherein the solving step further comprises dividing the loop variables into a list of time-invariant variables and a list of time-variant variables, identifying a range of times that encloses the active times of the time-invariant variables to generate an invariant range value, recursively ensuring that all prior loops that this loop depends on have been solved for all times in the invariant range, and for the time of interest, applying a solving step that simultaneously computes both the time-varying and the time-invariant variable values.

7. The method of claim 6, wherein dividing the loop and identifying the invariant range are accomplished at the same time as the loop identification and loop ordering.

8. The method of Claim 7, wherein dividing the loop and identifying the invariant range are accomplished simultaneously with the loop identification and loop ordering.

9. The method of claim 7, wherein the solving comprising implementing a multivariate optimization method that systematically explores potential variable values to optimize an objective function.

10. The method of claim 9, wherein the objective function for time-varying variables further comprises applying a sum of squares of the comparator values for each goal referenced by the variables, without duplication.

11. The method of claim 10, wherein the objective function for time-invariant variables further comprises applying a sum of squares of the applicable comparator values, summed, without duplication, over each time in the variables' active range for which the comparators are active.

12. The method of claim 11, wherein the solving further comprises iterating between solving the time-varying variables for each time in the invariant range, with tentative time-invariant values, and solving the time-invariant variables with tentative time-varying variable values in place.

13. The method of claim 12, wherein the comparator values are calculated every n-th time.

14. A system for solving complex systems of time-varying and time-invariant variables to optimize the satisfaction of goals, comprising:

means for explicitly identifying dependencies between variables;

means for analyzing the variable dependencies to identify groups of variables that are mutually interdependent to identify one or more loops;

means for ordering the loops, based on inter-loop dependencies, from most independent to most dependent; and

means, for any time of interest, for applying a solving step to each loop in turn, following the dependency ordering established above.

15. The system of claim 14 further comprising means for clearing previously computed solutions if anything on which those solutions depend is changed.

16. The system of claim 14, wherein loop identification and loop ordering are accomplished simultaneously.

17. The system of claim 14 further comprising means for deactivating one or more of a variable and a goal for specified time(s).

18. The system of claim 17 wherein each goal is specified as a computation ("comparator") yielding one or more measurements, which are optimally zero.

19. The system of claim 18 further comprising means for constraining a variable to be invariant over its active times, and wherein the solving means further comprises means for dividing the loop variables into a list of time-invariant variables and a list of time-variant variables, means for identifying a range of times that encloses the active times of the time-invariant variables to generate an invariant range value, means for recursively ensuring that all prior loops that this loop depends on have been solved for all times in the invariant range, and for the time of interest, means for applying a solving step that simultaneously computes both the time-varying and the time-invariant variable values.

20. The system of claim 19, wherein dividing the loop and identifying the invariant range are accomplished at the same time as the loop identification and loop ordering.

21. The system of Claim 20, wherein dividing the loop and identifying the invariant range are accomplished simultaneously with the loop identification and loop ordering.

22. The system of claim 21, wherein the solving means comprising a multivariate optimization system that systematically explores potential variable values to optimize an objective function.

23. The system of claim 22, wherein the objective function for time-varying variables further comprises applying a sum of squares of the comparator values for each goal referenced by the variables, without duplication.

24. The system of claim 23, wherein the objective function for time-invariant variables further comprises applying a sum of squares of the applicable comparator values, summed, without duplication, over each time in the variable's active range for which the comparators are active.

25. The system of claim 24, wherein the solving further comprises iterating between solving the time-varying variables for each time in the invariant range and solving the time-invariant variables with tentative time-varying variable values in place.

26. The system of claim 25, wherein the comparator values are calculated every n-th time.

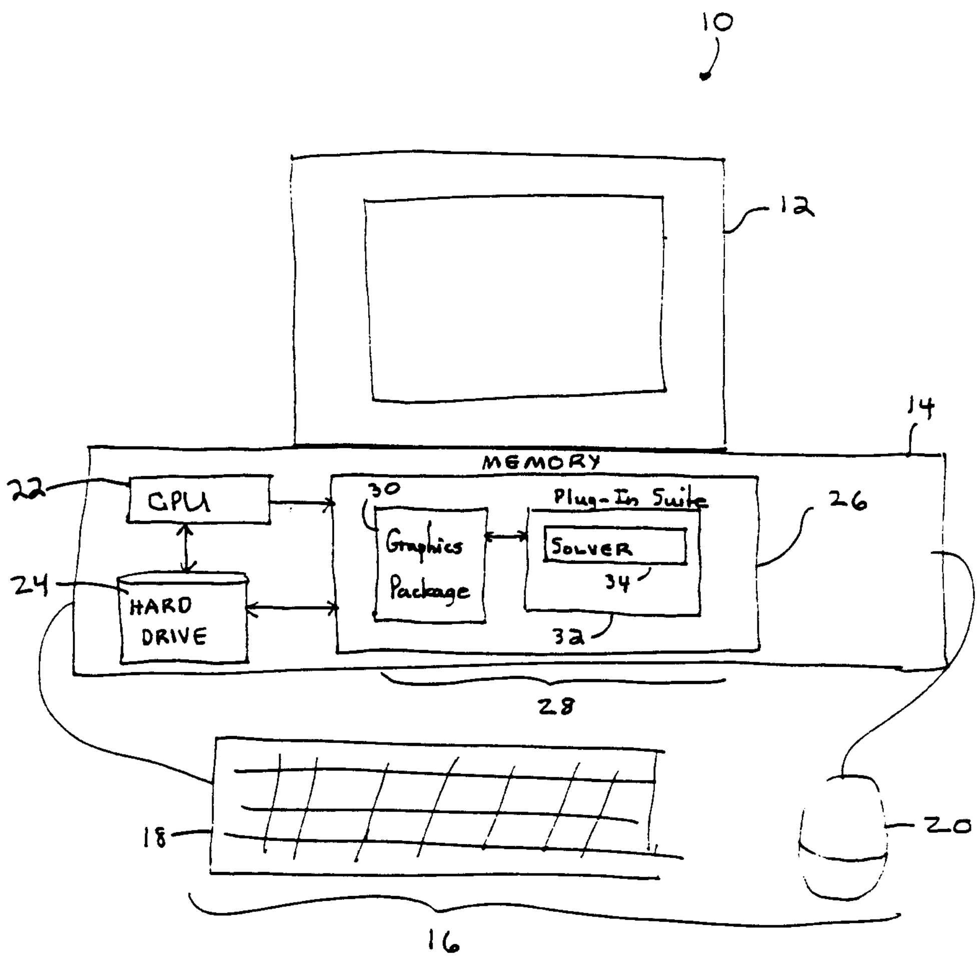

10
12
14
MEMORY
22
CPU
30
Graphics Package
Plug-In Suite
SOLVER
34
32
26
24
HARD DRIVE
28
18
20
16

FIGURE 1

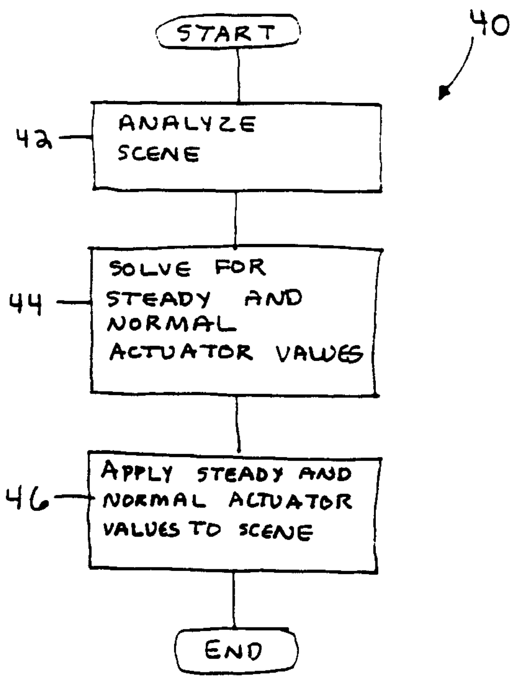

START
40
42
ANALYZE SCENE
44
SOLVE FOR STEADY AND NORMAL ACTUATOR VALUES
46
APPLY STEADY AND NORMAL ACTUATOR VALUES TO SCENE
END

FIGURE 2

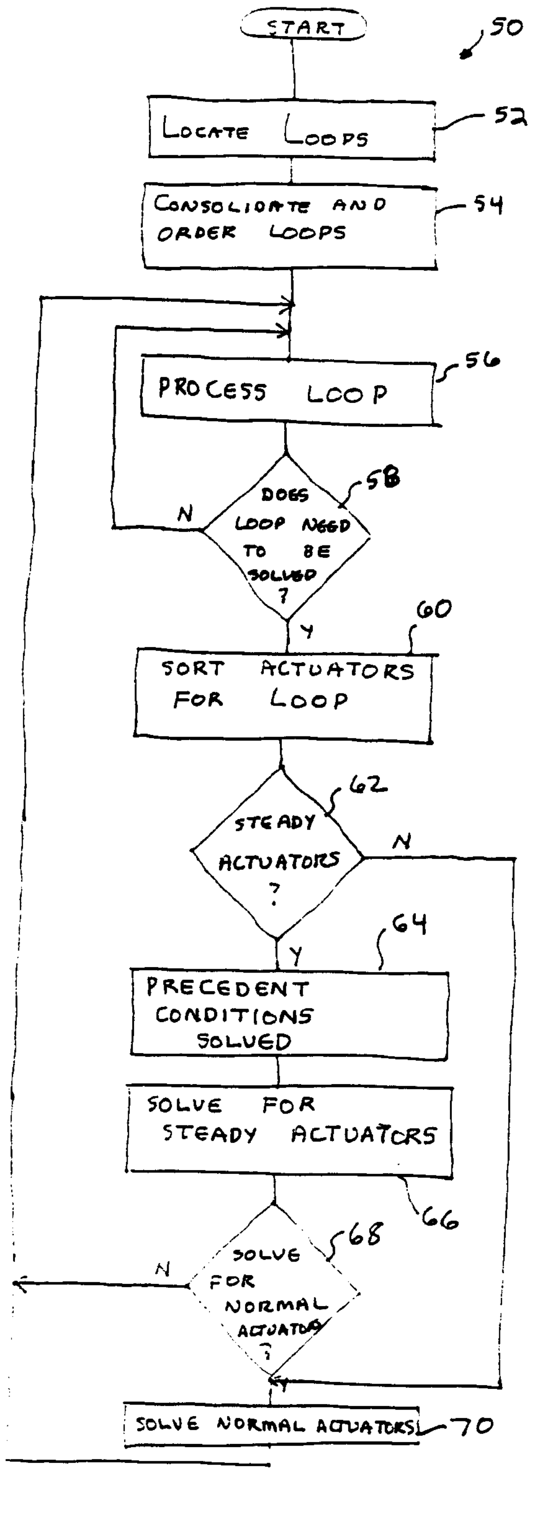

START
50
LOCATE LOOPS
52
CONSOLIDATE AND ORDER LOOPS
54
PROCESS LOOP
56
DOES LOOP NEED TO BE SOLVED ?
58
N
Y
SORT ACTUATORS FOR LOOP
60
STEADY ACTUATORS ?
62
N
Y
PRECEDENT CONDITIONS SOLVED
64
SOLVE FOR STEADY ACTUATORS
66
SOLVE FOR NORMAL ACTUATORS ?
68
N
Y
SOLVE NORMAL ACTUATORS
70

FIGURE 3

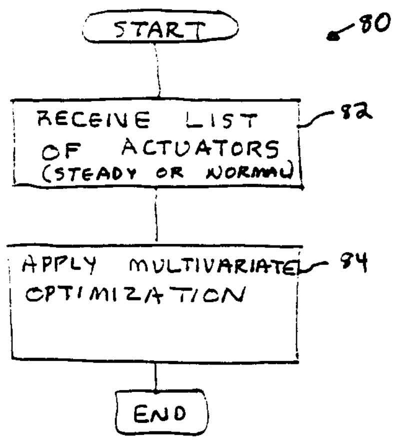

START
80
RECEIVE LIST OF ACTUATORS (STEADY OR NORMAL)
82
APPLY MULTIVARIATE OPTIMIZATION
84
END

FIGURE 4

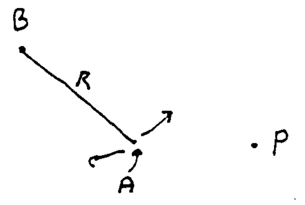


FIGURE 5

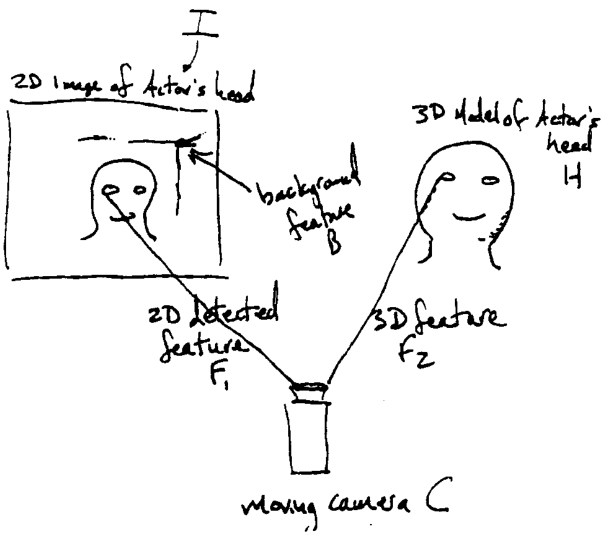


FIGURE 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/27849

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) :G06F 17/60
US CL : 705/7, 8, 9
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 705/7, 8, 9

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
WEST 2.0, CAS ONLINE, DIALOG, IEEE

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5,195,172 A (ELAD et al) 16 March 1993, see entire document.	1-26
A	US 5,521,814 A (TERAN et al) 28 May 1996, see entire document.	1-26
A	US 5,893,069 A (WHITE, Jr.) 06 April 1999, see entire document.	1-26

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance		
"E"	earlier document published on or after the international filing date	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed	"&"	document member of the same patent family

Date of the actual completion of the international search 13 DECEMBER 2000	Date of mailing of the international search report 09 JAN 2001
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	Authorized officer TARIQ R. HAFIZ James R. Matthews
Facsimile No. (703) 305-3230	Telephone No. (703) 305-9643