January 1990

Interactive Real-Time Articulated Figure Manipulation Using Multiple Kinematic Constraints

Cary B. Phillips
University of Pennsylvania

Jianmin Zhao
University of Pennsylvania

Norman I. Badler
University of Pennsylvania, badler@seas.upenn.edu

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-90-12.

Interactive Real-Time Articulated Figure Manipulation Using Multiple Kinematic Constraints

Abstract

In this paper, we describe an interactive system for positioning articulated figures which uses a 3D direct manipulation technique to provide input to an inverse kinematics algorithm running in real time. The system allows the user to manipulate highly articulated figures, such as human figure models, by interactively dragging 3D "reach goals." The user may also define multiple "reach constraints" which are enforced during the manipulation. The 3D direct manipulation interface provides a good mechanism for control of the inverse kinematics algorithm and helps it to overcome problems with redundancies and singularities which occur with figures of many degrees of freedom. We use an adaptive technique for evaluating the constraints which allows us to ensure that only a certain user-controllable amount of time will be consumed by the inverse kinematics algorithm at each iteration of the manipulation process. This technique is also sensitive to the time it takes to redraw the screen, so it prevents the frame display rate of the direct manipulation from become too slow for interactive control.

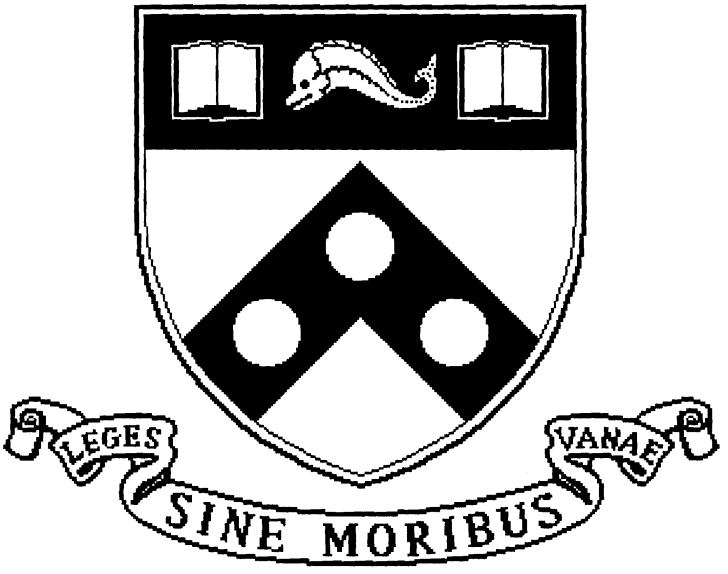
Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-90-12.

Interactive Real-Time Articulated Figure Manipualtion Using Multiple Kinematic Constraints

MS-CIS-90-12
GRAPHICS LAB 32

Cary B. Phillips
Jianmin Zhao
Norman Badler



University of Pennsylvania
School of Engineering and Applied Science
Computer and Information Science Department
Philadelphia, PA 19104-6389

1990

Interactive Real-Time Articulated Figure Manipulation Using Multiple Kinematic Constraints

MS-CIS-90-12
GRAPHICS LAB 32

Cary B. Phillips
Jianmin Zhao
Norman I. Badler

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104-6389

February 1990

Interactive Real-time Articulated Figure Manipulation Using Multiple Kinematic Constraints

Cary B. Phillips
Jianmin Zhao
Norman I. Badler

Computer Graphics Research Laboratory
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, Pennsylvania 19104-0389

Abstract

In this paper, we describe an interactive system for positioning articulated figures which uses a 3D direct manipulation technique to provide input to an inverse kinematics algorithm running in real time. The system allows the user to manipulate highly articulated figures, such as human figure models, by interactively dragging 3D "reach goals." The user may also define multiple "reach constraints" which are enforced during the manipulation. The 3D direct manipulation interface provides a good mechanism for control of the inverse kinematics algorithm and helps it to overcome problems with redundancies and singularities which occur with figures of many degrees of freedom. We use an adaptive technique for evaluating the constraints which allows us to ensure that only a certain user-controllable amount of time will be consumed by the inverse kinematics algorithm at each iteration of the manipulation process. This technique is also sensitive to the time it takes to redraw the screen, so it prevents the frame display rate of the direct manipulation from become too slow for interactive control.

Introduction

A major goal of the work in the Computer Graphics Research Lab at the University of Pennsylvania is to develop an interactive system for manipulating human figure models for static positioning, reach analysis, viewing assessment, and animation[1]. The models should behave according to user-defined terms, obeying degrees of freedom and joint limits. The system should allow users unskilled in computer programming to interactively manipulate the models effectively and intuitively under these constraints. We wish to be able to position the figure using general commands equivalent to such expressions as "keep the feet on the floor," "put the hand on the coffee cup," or "put the coffee cup in the hand and keep it level with the floor."

This facility is a part of **Jack**[6], a general purpose 3D modeling, manipulation, and animation system which is implemented on Silicon Graphics IRIS workstations. **Jack** is an interactive interface for visualizing and analyzing articulated figures. It provides a basic framework for command execution, graphic display, and interaction mechanisms by which users interact with a world of geometric objects.

Jack represents articulated figures through a language and data structure called **peabody**. **Peabody** represents figures composed of rigid segments connected by joints. The joints may have specific degrees of freedom and joint limits which are obeyed during manipulation.

Peabody requires that figures be tree-structured, but it defines them independently of the hierarchy so that figures may be rooted at any point in the tree. The root is not an intrinsic part of the figure definition: it is a user-changeable property. The root serves as the pivot point when the figure is manipulated as a whole. The underlying hierarchy of the figure is automatically recomputed whenever the user changes the root. Once the global placement of the root is set, the global placement of the remainder of the figure is completely defined

in terms of the segment dimensions and the local joint displacements. This mechanism allows the user to define the joints in a figure based on how he or she intends them to behave.

3D Direct Manipulation

The 3D direct manipulation facility in **Jack** allows the user to interactively manipulate figure positions and joint displacements[6]. The facility is built upon an operator which interactively manipulates general homogeneous transformations with a three button mouse and the keyboard. The manipulation operator is used throughout the **Jack** system whenever geometric information is required.

The direct manipulation operator is loop which repeatedly does the following:

1. read mouse coordinates and button status
2. convert mouse information into a 3D geometric transformation
3. apply transformation to the geometric environment (new figure location or joint displacement)
4. traverse object hierarchy to recompute global segment transformations
5. redraw the graphics windows

This loop continues until it is explicitly terminated or aborted by the user.

The design of this operator is based on the notion that it should be relatively easy for the user to manipulate gross geometric transformations when a lot of precision is not required. It should also be easy for the user to predict what motion of the mouse will cause the desired motion of the object.

The three mouse buttons control translation and rotation. The default operation is translation; rotation is activated by the control key on the keyboard. In translation, the left, middle, and right mouse buttons control translation along the x, y, and z axes, respectively. The user controls the motion by moving the mouse cursor along the line which the selected axis makes on the screen. Pairs of axes may be selected simultaneously to translate in a plane, in which case the transformation automatically moves to the point in the selected plane which lies underneath the mouse cursor. A 3D graphical translation icon located at the origin of the object being manipulated illustrates the selected axes and enabled directions of motion. This technique is related to Bier's "skitters" [3].

When the user holds down the control key on the keyboard, the transformation becomes rotation, in which the left, middle, and right mouse buttons control rotation around the x, y, and z axes, respectively. Only one axis may be selected at a time. The axis is illustrated by a graphical "wheel" icon which describes the origin and direction of the axis. The user controls the rotation by moving the cursor around the perimeter of the rotation wheel, causing the transform to rotate around the axis. This is analogous to turning a crank by grabbing the perimeter. Direct manipulation of rotations by "stirring" without the wheel display were used Wein[8].

Inverse Kinematics for Multiple Goals

This type of direct manipulation is fairly easy to use, although it can be a tedious and ineffective way of manipulating figures as complex as the human body. Inverse kinematics techniques used in robotics [5] are useful except that the figures with which we are concerned are highly redundant so that goal-directed positioning tasks may have an infinite number of solutions.

We have built a more powerful positioning facility which uses an iterative optimization technique, allowing us to solve for figure positions which satisfy multiple simultaneous kinematic constraints [10]. We define a kinematic constraint in terms of a goal coordinate frame, an end effector coordinate frame, and a set of joints which control the end effector. The user describes the set of joints by selecting a "starting joint." The joint set then consists of the chain of joints between the starting joint and the end effector. We sometimes call this a "reach" constraint because the application easiest to visualize is a reaching human arm, with the fingertips as the end effector. However, it can be used on any joint chain in the figure, not just at the extremities. We sometimes refer to the entire constraint as a "goal" since it represents something to be achieved. The goal will be achieved when we arrive at a set of joint angles which place the end effector at the goal, according to some user-controllable criteria.

We begin by phrasing the constraint in terms of a minimization problem. For a specific reach constraint, the position and orientation of the end effector in space is functionally dependent on the joint angles $\theta_1, \theta_2, ..., \theta_n$:

$$\mathbf{e} = \mathbf{e}(\theta_1, \theta_2, ..., \theta_n)$$

where n is the total number of degrees of freedom of the joint chain. The value of the function $\mathbf{e}$ is a matrix which describes both position and orientation. Associated with each goal there is a characteristic vector function $\mathbf{V}$ such that the goal is met if and only if the function applied to the end effector yields the zero vec-

tor:

$$\mathbf{V}(\mathbf{e}) = \mathbf{0}$$

Notice that $\mathbf{V}$ is a vector function of spatial arguments. For example, for a positional goal with fixed position $\mathbf{x}$, $\mathbf{V}$ is the simple euclidean distance function:

$$\mathbf{V}(\mathbf{e}) = \mathbf{e}_p - \mathbf{x}$$

where $\mathbf{e}_p$ is the positional component of $\mathbf{e}$. The orientational component is ignored.

We, like others[9], have developed several types of characteristic functions:

orientation Only the orientation of the end effector is significant.

weighted position and orientation The position and orientation of the end effector are both significant, according to an arbitrary weighting factor. The default weight makes 5 degrees of angular displacement approximately equivalent to one centimeter of euclidean distance.

line The end effector is constrained so that its position must line along a specific line in space.

direction The end effector is constrained so that it "aims" a reference vector in its own coordinate frame towards the origin of the goal.

plane The end effector is constrained so that its origin lies in a specific plane in space.

Once the characteristic functions are set up, the problem is then to solve for $\theta_1, \theta_2, ..., \theta_n$ so that

$$\mathbf{V}(\mathbf{e}(\theta_1, \theta_2, ..., \theta_n)) = 0$$

where $\mathbf{V}$ may contain several goals instead of just one. At each iteration k, we compute the joint angle vector Θ_k by first computing the partial derivative of $\mathbf{V}$ with respect to each joint angle θ_j and use this as a first-order approximation of $\mathbf{V}$:

$$\mathbf{V} = \mathbf{V}(\Theta_k) + \frac{\partial V_i}{\partial \theta_j}(\Theta - \Theta_k)$$

We then compute the pseudo-inverse of $\frac{\partial V_i}{\partial \theta_j}$ to determine $\delta\Theta$:

$$\delta\Theta = \frac{\partial V_i}{\partial \theta_j}^{+} \mathbf{V}(\Theta_k)$$

This process continues until either the end effectors are within some user-controllable tolerance of their goals, or successive iterations do not decrease the distance towards the goal.

Although this is a typical root-finding problem for algebraic equations, there are some special requirements:

- the joint angle variables $\theta_1, \theta_2, ..., \theta_n$ are bounded by upper and lower limits. Solutions which are not in this region are deemed not feasible.
- The algorithm should converge from any feasible initial configuration.
- When the system is underconstrained, the redundancies should be resolved in some acceptable manner.
- When the system is overconstrained, the solution should be a close as possible.
- The algorithm should be fast enough to be used interactively.

The Newton-Raphson method is very powerful, but it is not globally convergent so its behavior depends on the initial guess of the solution. To overcome this defect, we use a hybrid method proposed by Powell.[7] In addition to $\mathbf{V}$, Powell also considers the scalar function

$$f = \mathbf{V}^2$$

Clearly, $\mathbf{V}(\theta_1, \theta_2, ..., \theta_n) = \mathbf{0}$ if and only if $f(\theta_1, \theta_2, ..., \theta_n) = 0$. Powell uses the Newton-Raphson method whenever that method decreases the function f enough. Otherwise, the gradient (steepest descending) method is used, although this may end up with a local minimum. We have extended this method to consider the limits on the variables θ. Currently, we consider the special case

$$\text{lower\_limit}_i \leq \theta_i \leq \text{upper\_limit}_i$$

and use the projection method for linear constraints.

We use a pseudo-inverse method to compute the Newton-Raphson solution since the Jacobian matrix $\frac{\partial V_i}{\partial \theta_j}$ is not in general invertible and typically is not even a square matrix. The solution complexity for the pseudo-inverse of the Jacobian is then $O(n^2)$, where n is the number of the degrees of freedom. But for the multiple goal problem, if the number of goals is comparable to the number of degrees of freedom n, the Jacobian would be $O(n) \times O(n)$ and the complexity therefore becomes $O(n^3)$.

So in the multiple goal case, we chose to use a variable metric method to minimize the function f with linear constraints for θ's. In this method, inversion of the matrix is not computed explicitly. Its approximation is improved from each iteration to the next. Each iteration needs $O(n^2)$ operations. It is super-linear convergent. A detailed explanation of this algorithm is available [10].

Positioning with Inverse Kinematics

The ability to solve multiple inverse kinematic constraints is very powerful, and our implementation is fairly computationally efficient, but by itself it has some severe limitations. One basic interface for the inverse kinematic positioning facility is a command-oriented specification of the reach parameters. The user selects the goal, the type of reach (the characteristic function), the end effector, the starting joint, and the other numeric parameters which control the reach. This is done successively for each kinematic constraint. After the complete specification of all constraints, the inverse kinematics algorithm is invoked to solve for each joint angle, after which the figure assumes its new position.

This interface fails for several reasons. The positioning tasks for which this facility is used are usually underconstrained. The prototypical example is the reaching human arm, where the position of the elbow is not uniquely specified by a positional goal for the hand or fingers. The problem becomes much more complex for position tasks involving more degrees of freedom, such as a reach with a joint chain extending from the hand to the waist.

The algorithm also suffers from problems of local minima, since the initial guess of the solution comes from the current configuration of the joints. This means that it may fail to arrive at a solution even when one does exist, or that the solution at which it arrives is not really the one the user intended. The only feasible approach in this type of interface is for the user to reissue the reach command with more constraints to reduce the number of redundancies.

Interactive Methodology

There are several possibilities for overcoming the problems with redundancies and local minima. One is to incorporate more information into the objective function, modeling such factors as strength, comfort, and agent preference [11]. This is an important addition, although it adds significantly to the computational complexity of the goal solving procedure. Our technique is to provide the positional input to the inverse kinematics algorithm with the 3D direct manipulation system. We allow the user to interactively "drag" goal positions and have the end effector follow. In this case, the geometric information obtained by the mouse at each iteration of the manipulation process is applied to the goal position of a reach, and the inverse kinematics algorithm is called to solve the goals before the graphics windows are redrawn.

This dragging mechanism is a modified version of the basic direct manipulation scheme. After selecting the parameters of the reach, the manipulation procedure proceeds as follows:

1. read mouse coordinates and button status
2. convert mouse information into a 3D geometric transformation
3. apply transformation to the placement of the goal
4. invoke inverse kinematics positioning algorithm
5. redraw the graphics windows

The inverse kinematics procedure is invoked not just once, but at every screen refresh during the interactive manipulation.

This a very effective and efficient tool for manipulation for several reasons. Because of the incremental nature of the interactive manipulation process, the goals never move very far from one iteration to the next. Therefore, the initial guess for the inverse kinematics algorithm is almost always very good, making the algorithm effectively very computationally efficient. The algorithm still suffers from problems of local minima, but since the user can drag the end effector around in space in a well-defined and easy to control way, it is relatively easy to overcome these problems by "stretching" the figure into temporary intermediate configurations to get one part of the figure positioned correctly, and then dragging the end effector itself into the final desired position.

A common example of this dragging technique involves the elbow. The user may initially position the hand at the proper place in space but then find that the elbow is too high. If this is the case, the user can extend the hand outwards to drag the elbow into the correct general region and then drag the hand back to the proper location. This is illustrated in Plates 1 and 2. Plate 1 illustrates an awkward position of the elbow during a reach. Plate 2 shows a better elbow position which was achieved by interactively dragging the hand out and then back. The trace shows the 3D path along which the hand was dragged.

Another effective feature of the direct manipulation interface is the use of orientation constraints, particularly the weighted combination of position and orientation. In this case, the orientation of the goal is significant as well as the position, so the user may manipulate segments in the interior of the reach chain by twisting the orientation of the goal and end effector. This is especially helpful because of the difficulty the user encounters in visualizing and numerically describing rotations which will achieve a desired orientation. The above example of the elbow position may be handled this way,

too. By twisting the desired orientation of the hand, the interior of the arm can be rotated up and down while the hand remains the the same location. This achieves in real-time a generalization of the "elbow circle" positioning scheme implemented by Korien. [4]

Plates 3 and 4 show a sequence of rotating the arm from the hand, with the rotation wheel. Plate 5 shows a rotation of both the arm and the torso.

Manipulation with Constraints

The nature of the 3D direct manipulation mechanism allows the user to interactively manipulate only a single element at a time, although most positioning tasks involve several parts of the figure, such as both feet, both hands, etc. The interactive reach described above manipulates only a single chain of the figure at one time.

In addition to interactively dragging a single end effector, the user may define any number of kinematic "reach constraints" which are goals of any objective type to be enforced as the figure is manipulated using any of the other manipulation tools. By first defining multiple constraints and then manipulating the figure, either directly or with the dragging mechanism, the user may enforce complex positioning restrictions.

This mechanism involves another slight modification to the direct manipulation loop:

1. read mouse coordinates and button status
2. convert mouse information into a 3D geometric transformation
3. apply transformation to the geometric environment (new figure location or joint displacement)
4. traverse object hierarchy to recompute global segment transformations
5. invoke inverse kinematics positioning algorithm to solve multiple goals
6. redraw the graphics windows

Step #4 may cause the end effectors to move away from their goal positions. The inverse kinematics algorithm in step #5 repositions the joints so the goals are satisfied.

Plate 6 shows a posture achieved by interactively manipulating the figure under the influence of four reach constraints constraining the feet to the floor and the hands to the toes.

We are primarily interested in the interactive nature of the system. The user must have the feeling of real-time control over the figures. A slow screen update rate is detrimental to this sense of interactive control. Unfortunately, the inverse kinematics algorithm can be fairly time consuming when there are several constraints. The lag time between the motion of the mouse and the ensuing motion of the objects makes the manipulation process difficult to control. To alleviate this problem, we limit the amount of time which can be consumed by the inverse kinematics algorithm at each interactive iteration.

The inverse kinematics algorithm is iterative, and it converges monotonically, so at each iteration the end effectors move closer to the goals. We exploit this property and accept an intermediate solution if the entire solution cannot be computed quickly enough. Rather than limiting the number of iterations, we limit the amount of time consumed. We do this by recording the time at which the algorithm begins[1], and then checking the current time at the beginning of each iteration. If the time limit has expired, we terminate the algorithm and accept the current configuration. The direct manipulation process then proceeds with the next interactive iteration.

This has an interesting effect on the "feel" of the manipulation. With the time limit set properly, the frame rate never deteriorates beyond several frames per second even with several constraints, so the user never looses the sense of interactive control. However, the end effectors move more slowly towards their goals. For example, with constraints on the hands and feet, the user may quickly yank the figure away from its current location, and the arms and legs will gradually drift back in the direction of their goals. This fills the "dead time," when the user is just looking at the screen, with useful computation.

We have also developed an adaptive technique for making this time limit sensitive to the amount of time consumed at each frame by the drawing of the graphics windows. This works well since when there are many large, complex geometric objects, much of the time consumed by the manipulation loop is spent in drawing the graphics windows. This sensitivity means that the amount of time allotted to the inverse kinematics algorithm is automatically decreased. We implement this by keeping a record of how much time is consumed each time the screen is drawn. This timing information is only approximate.

Performance and Examples

The interactive performance of the inverse kinematics positioning facility depends of course on the speed of the workstation on which it runs, but it runs well on any

[1] This information is available in 60th's of seconds

of the Silicon Graphics IRIS 4D line of workstations, including the Personal IRIS. We give some approximate timing values here for an IRIS 4D-70GT. The human figure model we manipulate consists of 653 wireframe vectors, or 378 shaded polygons, with 30 segments, 29 joints, and 53 total degrees of freedom. The ordinary direct manipulation interface without inverse kinematics displays the wireframe model at approximately 25 frames per second. The shaded model displays at 15 frames per second.

Under the influence of a single constraint consisting of 7 degrees of freedom, the rate is approximately 10 fps when the goal is reachable, slightly less when the goal is not achievable.

Under the influence of 4 simultaneous constraints consisting of 7 degrees of freedom each (one for each arm and leg), the rate never deteriorates beyond 3-4 fps.

We set the default value of the iteration time limit to be 0.1 seconds, and we have found empirically that this value works quite well. The value is controllable by the user, but there is actually little need to adjust it.

Future work

The current implementation of our kinematics algorithm is purely geometric. It uses no other criteria to evaluate the acceptability of a goal solution other than the joint angles, subject to the joint limits. We are currently developing strength and comfort models to encorporate into the objective functions. We are also developing collision detection and avoidance capabilities.

Acknowledgements

This research is partially supported by Lockheed Engineering and Management Services (NASA Johnson Space Center), NASA Ames Grant NAG-2-426, FMC Corporation, Martin-Marietta Denver Aerospace, NSF CER Grant MCS-82-19196, and ARO Grant DAAL03-89-C-0031 including participation by the U.S. Army Human Engineering Laboratory.

References

[1] Norman I. Badler, "Artificial Intelligence, Natural Language, and Simulation for Human Animation" in *State-of-the-Art in Computer Animation*, N. Magnenat-Thalmannn and D. Thalmann (eds.), Springer-Verlag, 1989, pp. 19-31.

[2] Norman I. Badler, Kamran Manoochehri, and Graham Walters, "Articulated Figure Positioning By Multiple Constraints", *Computer Graphics and Applications*, Vol. 7, No. 6, June, 1987.

[3] Eric Allan Bier, "Skitters and Jacks: Interactive Positioning Tools," In *Proceedings of 1986 Workshop on 3D Interactive Computer Graphics*, (Chapel Hill, NC, October 23-26, 1986), ACM, New York, 1987.

[4] James Korien, *A Geometric Investigation of Reach*, MIT Press, 1985.

[5] Richard P. Paul, *Robot Manipulators: Mathematics, Programming, and Control*, MIT Press, 1981.

[6] Cary B. Phillips and Norman I. Badler, "Jack: A Toolkit for Manipulating Articulated Figures" *Proceedings of ACM/SIGGRAPH Symposium on User Interface Software*, Banff, Alberta, Canada, 1988.

[7] M.J.D. Powell"A Hybrid Method for Nonlinear Equations", in *Numerical Methods for Nonlinear Algebraic Equations*, edited by Philip Rabinowitz, Gordon and Breach Science Publisher, 1970.

[8] M. Wein et. al. "Graphics Interactions at NRC", *SIGGRAPH Video Review Issue #4*, ACM, 1981.

[9] Andrew Witkin, Kurt Fleischer, and Alan Barr, "Energy Constraints on Parametrized Models," *Computer Graphics 21*, No. 3, 1987.

[10] Jianmin Zhao and Norman I. Badler, "Real Time Inverse Kinematics With Joint Limits and Spatial Constraints", Technical Report MS-CIS-89-09, University of Pennsylvania.

[11] Philip Lee, Norman Badler, Cary Phillips, and Ernest Otani, "The Jack Interactive Human Model", in *Proceedings of the First Annual Symposium on Mechanical System Design in a Concurrent Engineering Environment*, Iowa City, Iowa, October 1989.

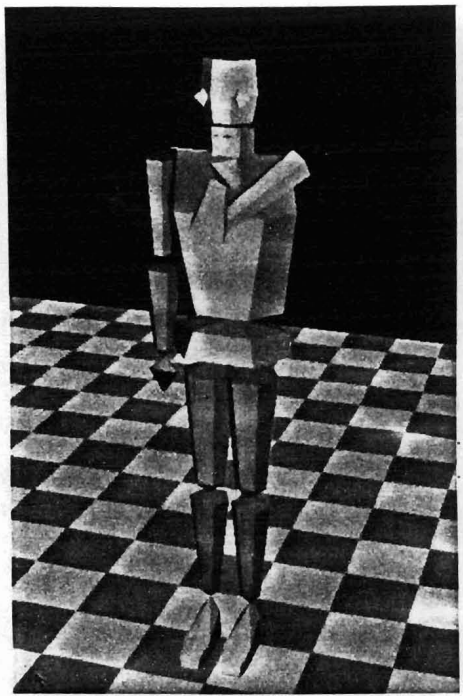


PLATE 1

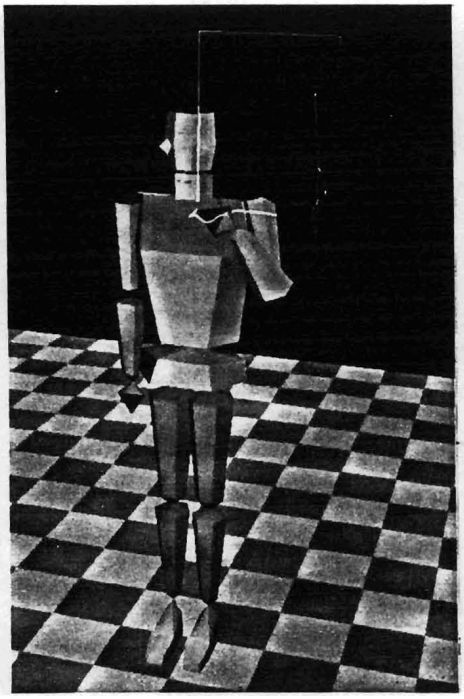


PLATE 2

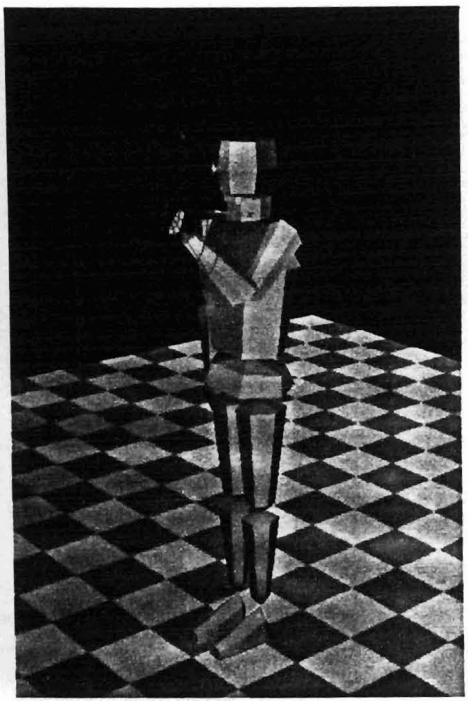


PLATE 3

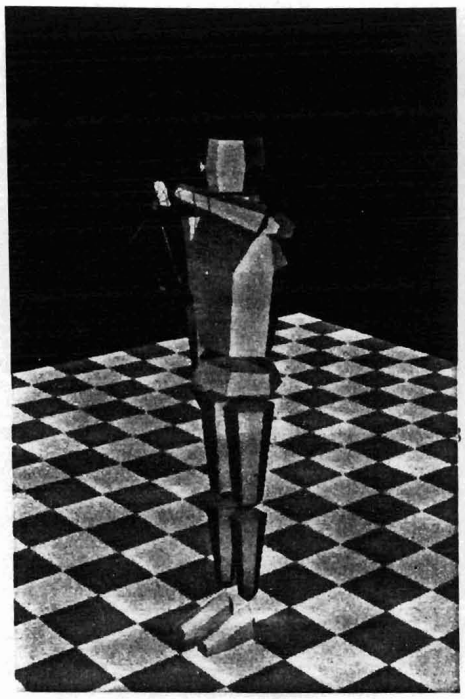


PLATE 4

PLATE 5

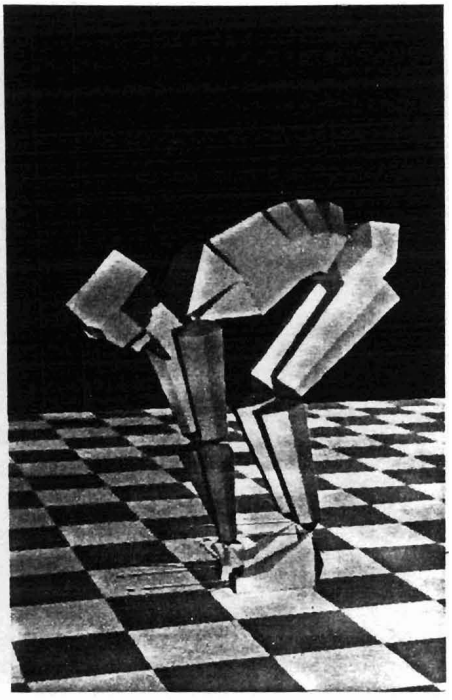


PLATE 6

Plate 1 illustrates an awkward position of the elbow during a reach. Plate 2 shows a better elbow position which was achieved by interactively dragging the hand out and then back. The trace shows the 3D path along which the hand was dragged. Plates 3 and 4 show a sequence of rotating the arm from the hand, with the rotation wheel. Plate 5 shows a combined arm and torso rotation. Plate 6 shows a posture achieved by interactively manipulating the figure under the influence of four reach constraints constraining the feet to the floor and the hands to the toes.