January 1992

Generating Human Motion by Symbolic Reasoning

Moon Ryul Jung
University of Pennsylvania

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

Generating Human Motion by Symbolic Reasoning

Abstract

This paper describes work on applying AI planning methods to generate human body motion for the purpose of animation. It is based on the fact that although we do not know how the body actually controls massively redundant degrees of freedom of its joints and moves in given situations, the appropriateness of specific behavior for particular conditions can be axiomatized at a gross level using commonsensical observations. Given the motion axioms (rules), the task of the planner is to find a discrete sequence of intermediate postures of the body via goal reduction reasoning based on the rules along with a procedure to discover specific collision-avoidance constraints, such that any two consecutive postures are related via primitive motions of the feet, the pelvis, the torso, the head, the hands, or other body parts. Our planner also takes account of the fact that body motions are continuous by taking advantage of execution-time feedback. Planning decisions are made in the *task space* where our elementary spatial intuition is preserved as far as possible, only dropping down to a *joint space formulation* typical in robot motion planning when absolutely necessary. We claim that our work is the first serious attempt to use an AI planning paradigm for animation of human body motion.

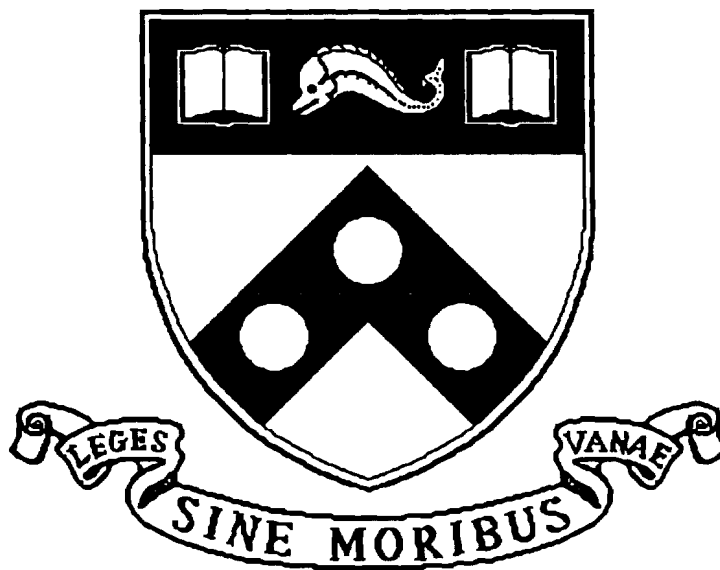## Comments

Generating Human Motion By Symbolic Reasoning

Moon Ryul Jung
Norman I. Badler

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

January 1992

Generating Human Motion By Symbolic Reasoning

Moon Ryul Jung and Norman I. Badler

Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104-6389

Abstract

This paper describes work on applying AI planning methods to generate human body motion for the purpose of animation. It is based on the fact that although we do not know how the body actually controls massively redundant degrees of freedom of its joints and moves in given situations, the appropriateness of specific behavior for particular conditions can be axiomatized at a gross level using commonsensical observations. Given the motion axioms (rules), the task of the planner is to find a discrete sequence of intermediate postures of the body via goal reduction reasoning based on the rules along with a procedure to discover specific collision-avoidance constraints, such that any two consecutive postures are related via primitive motions of the feet, the pelvis, the torso, the head, the hands, or other body parts. Our planner also takes account of the fact that body motions are continuous by taking advantage of execution-time feedback. Planning decisions are made in the *task space* where our elementary spatial intuition is preserved as far as possible, only dropping down to a *joint space formulation* typical in robot motion planning when absolutely necessary. We claim that our work is the first serious attempt to use an AI planning paradigm for animation of human body motion.

1 Introduction

The problem of animating movement of the human body in a realistic manner is a challenging goal of computer graphics researchers. The animation of the human body has applications in a wide variety of fields such as entertainment (e.g., cartoon), ergonomic studies (e.g., the ability of people to do work in a given environment), and computer-aided design of human workspaces (e.g., design of interiors for cars, planes, and space vehicles).

One of the issues in human figure animation is to liberate the designer from the technical details of animation specification and control. The issue is how to generate a sequence of motions needed to achieve a given task or goal. This paper advocates generating a sequence of motions for a given goal by applying AI planning methods and devises a human motion planning method called *posture planning*. AI planning methods, generally speaking, assume that (1) a given agent has a repertoire of actions that it can carry out, (2) an action has conditions under which the action can be applied and yields effects which may be conditional, (3) a goal is a result of a sequence of actions. The task of the planner is to find a sequence of actions that yields a given goal so that the actions satisfy given action rules and constraints. Applying AI planning method to motion generation is based on the fact that although we do not know how the body controls massively redundant degrees of freedom of its joints and moves in given situations, (1) a list of basic body motions can be identified and (2) the appropriateness of a motion or a set of motions for given conditions can be axiomatized at a gross level using commonsensical observations. Given the motion axioms (rules), a planner can be designed to find a discrete sequence of intermediate postures of the body via goal reduction reasoning based on the motion rules along with a procedure for maintaining constraints, e.g., collision-avoidance constraints and the body balance constraint.

In summary, the posture planner for task level motion goals works on the following principles:

1. Devise motion rules that "axiomatize" motions by means of their subgoals and effects. In particular, the motion rules are supposed to capture combining two parallel motions, e.g., stretching a hand to the ground while lowering the body.

2. Devise posture planning heuristics for selecting and sequencing primitive motions to achieve the goal condition of the end effector such as a foot, the head, and hand(s).

The input to the posture planner is a goal condition to be achieved by the agent. The output is a chart of overlapping primitive motions and constraints of the body parts needed to achieve the input goal condition.

Only the geometric aspects of gross motion are considered. The search-based route planning for walking (see Ridsdale 1986) is not considered. We are concerned with approaching the task region and performing the given task, e.g., approaching the table to pick up an object on it.

As an example, consider an agent who stands in front of a table (figure 1) and is given a goal of *picking up* the block under the table (figure 2). From the nature of the goal, the goal condition of the end effector - a hand - can be determined. But the goal condition of the end effector is not sufficient to initiate posture planning. The agent needs to figure out goal

conditions and constraints of the whole body that would enable the end effector to *reach*[1] the block without colliding with the table.
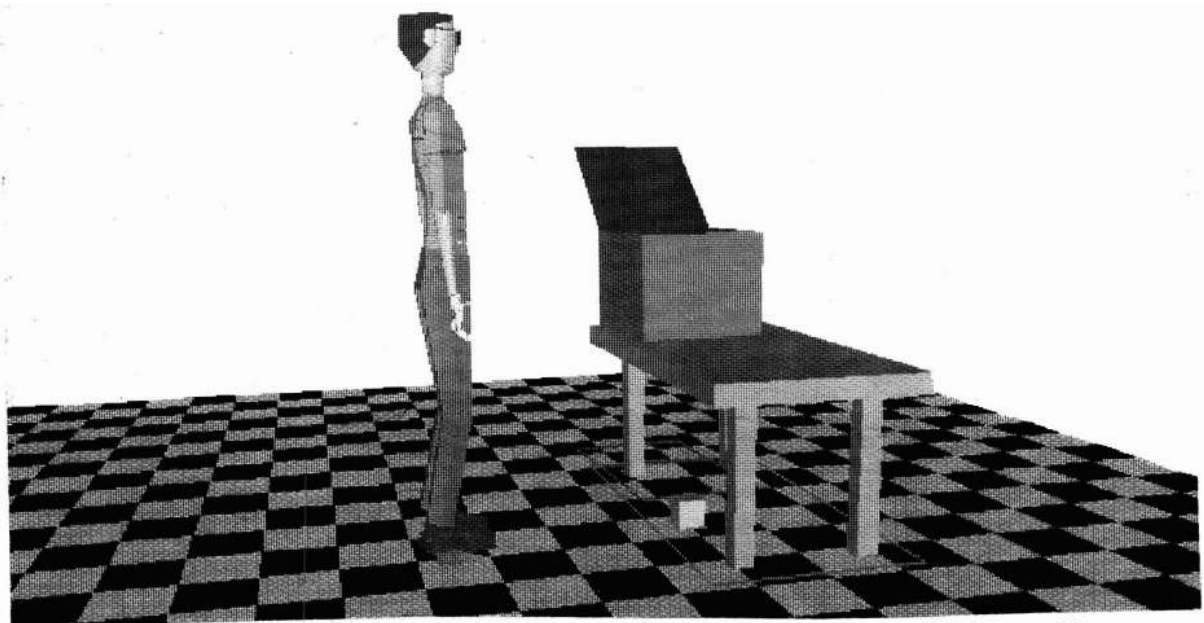
Figure 1: The agent in front of the table. A small block is under the table.
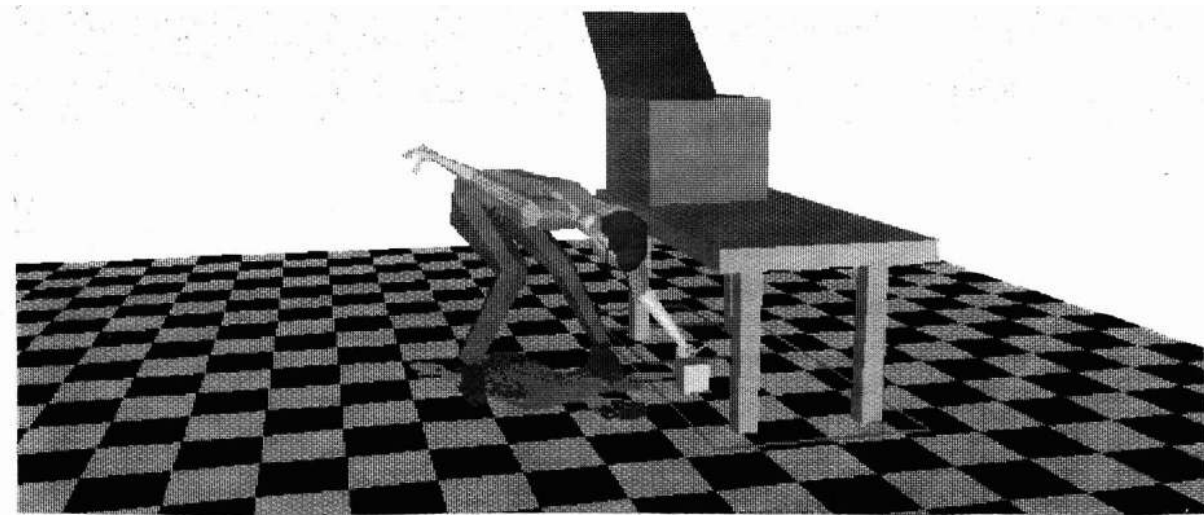
Figure 2: Reaching and grasping the block under the table.

In our approach, the conditions on the whole body are specified by goal conditions and constraints of important body parts, e.g., the feet, the pelvis, the torso, and the head. To do so, the agent would (1) *look at* the block to figure out the goal conditions and constraints of

[1]The italicised verbs correspond to primitive motions of a human figure.

important body parts that are necessary to *grasp* the block with a hand. The agent would generate collision avoiding constraints of the end effector, the hand, such that the end effector should be under the table and horizontally away from the left and right legs of the table. The forward orientation of the body at the goal region is set so that the target object, the block, is in front of the body, if possible. Since the forward orientation of the body does not affect the ability of the hand to reach the block too much, but can be used to prevent the head from colliding with the table, the forward orientation of the body at the goal region is deviated from the default orientation. Given the positions and orientations of the important body parts, the agent would generate a sequence of primitive motions to locate the body in those positions and orientations.

It is somewhat understandable that computer graphics researchers have not been overly successful in exploring AI planning approaches to generate human body motions for the purpose of animation. AI planning methods were motivated to create intelligent self-governing robots (Fikes and Nilson 1971; Munson 1971; Tate 1975, 77; Sacerdoti 1977; Waldinger 1977; McDermot 1982; Allen 1983, 84; Wilkins 1984; Dean 1987; Agre 1987; Chapman 87; Schoppers 1989). But there is still a gap between the prevalent symbolic planning paradigm of AI planning and *robot motion* planning which takes seriously the geometric and physical nature of robot motions. However, the problem is not with the symbolic paradigm itself, but the lack of empirical work linking symbols with their physical contents. From the view point of animation, animated agents need to have a planning capability in order to liberate animation designers from specifying tedious details. For example, over the years, we have developed an extensive model-based human body animation system (Phillips & Badler 1991). The human body model of the system is anthropometrically realistic and embodies kinematic and certain dynamic properties, so that agents of different builds and strengths can be animated to perform tasks such as *grasp, lift, look at, bend, stand up, sit down* etc. However, sooner or later, it became obvious that we wanted the animated agent to have a planning ability to select appropriate motions for given situations and sequence them in order to achieve a given goal. To achieve this goal, we can apply an AI planning paradigm by providing domain specific motion rules and domain specific reasoning strategies.

The present study can be claimed as the first serious attempt to apply *planning* to animation of the human body. A preliminary result of it is reported in Jung et. al (1991). Ridsdale's *Director's Apprentice* (1986) developed a character path animation system that accepts rule-based specification. But the rules consist of condition-action pairs and the animation system uses them only for control purposes, that is, when to activate what actions, but not for planning purposes. What Zeltzer (1985) and Esakov and Badler (1990) call *task-level* control is actually task-level programming, not planning. Reynolds (1987) proposed *behavioral animation* which is driven by control heuristics reactive to environmental situations. Renault and Thalman (1990) described a rule-based system which consists of a set of condition-action pairs plus a control structure over it. The animation is generated as the controller decides what action to activate based on the current situation. But still, there is no notion of planning in this paradigm; condition-action pairs are merely incremental heuristics applied frame by frame with no planning. Lee et. al (1990) developed a system that employs low-level reactive decision making based on the comfort level of the current motion. We go further and advocate goal-

4

directed planning in addition to reactive responses.

In devising a goal-directed planning, it is the contention of this paper that motion planning should use the *task space formulation* as far as possible and the *joint space formulation* should be used only when it is absolutely needed. As in the robot motion planning field, most animation methods formulate the animation problem with respect to the *joint space* of the body in which elementary geometric intuition is no longer preserved. The reason is that the configuration of a body at a given moment is determined by a sequence of joint angles, that is, motion of the body is ultimately represented in terms of joint space coordinates. For instance, consider the constraint optimization method of Witkin and Kass (1988). In this method, the animator is supposed to specify the initial posture (frame) and the final posture of a desired motion. Constraint optimization views the given postures as constraints and tries to optimize some criteria such as energy to determine in-between postures. But the very problem of finding the final posture of the body challenges us, especially for the human task animation, because the input goal typically specifies only the goal condition of the end effector, e.g., a hand.

The joint space is highly abstract and operations in it cannot be given intuitive meaning. The joint space formulation loses task space information that is useful for motion planning. For example, elementary task space spatial notions, e.g., *left, front, up*, and *down* would be useful for spatial maneuvers, as indicated by natural language instructions that use such spatial relations to specify human motion. Also, in the joint space it is difficult to define primitive motions of the body over which the planner does reasoning. Because motion primitives are essential to the AI planning paradigm, the task space formulation which allows us to form motion primitives with respect to the task space is desirable. Indeed, some researchers advocated the task space formulation of motion problems (Khatib 1986, Buckley 1989, Maciejewski 1985; Barraquand, Langlois & Latombe 1989; Breen 1989). In the task space formulation, motion of the body is specified and controlled in terms of task space measurements. The transformation into the joint space is postponed until motion planning decisions that are better made in the task space have been made. The task space formulation is desirable because it gives more clues for moving the body than does the joint space formulation. Moreover, the potential field method is faster than any other method of motion planning of articulated bodies and is capable of handling bodies with many degrees of freedom like the human body (Barraquand, Langlois, and Latombe 1989).

The goal-directed symbolic reasoning with respect to the task space helps reduce the complexity of motion planning. The robot motion planning algorithms are inherently exponential in the degrees of freedom of the robot manipulator. There are some practical robot motion planning algorithms (Brooks 1983a, Lozano-Perez 1987) for robots of five or six degrees of freedom. Those algorithms are obtained by some decomposition techniques that decompose the motion of the robot into submotions with three or four the number of degrees of freedom. The goal-directed symbolic reasoning can be viewed as generalization of decomposition techniques. In our approach, commonsensical motion planning with respect to the task space is used to determine motions of body parts needed to achieve a given goal. For instance, the planner may say, " bend the pelvis while lowering it to grasp an object on the ground".

However, the task space formulation of the problem has its own problem. The posture planner must also check if a set of generated motions, together with motion constraints such as

5

the body balance and collision avoiding constraints, can be achieved simultaneously. It is done by a *body positioner*, which tries to find a body posture that achieves a given set of motions and constraints. The body positioner is usually implemented by an inverse-kinematics algorithm. In particular, we use a general inverse-kinematics algorithm devised by Zhao (1989). It tries to find a sequence of joint angles that satisfy a given set of task space conditions of body parts. It is formulated as a nonlinear optimization problem. It treats task space conditions as objective functions or potential energy functions of joint angles and finds joint angles that minimizes a global objective function, a weighted sum of individual objective functions. The optimization algorithm searches for joint angles using the gradient of the global objective function. The problem is that the algorithm is extremely local. That is, it stops when the gradient of a global potential function is zero even if the gradients of individual potential functions are not zero. In other words, a given set of multiple goals is considered to be satisfied even when individual goals are not. This is called the *local minimum* problem. The local minimum situation is frequently reached in the case of a human figure, because its joints are massively redundant. We solve this problem also by employing reasoning explicitly so that local minimum situations are predicted and thus avoided. How to predict and avoid local minimum situations will be briefly discussed at the end of the paper.

2 Primitive Motions and Motion Rules

2.1 Defining Primitive Motions

In order to use an AI planning paradigm, we need to first define primitive motions over which the planner can do reasoning. To do so, the body structure is first defined. In our work on human body animation, the human body is modelled as a linkage of polyhedral segments. Our human figure model is anthropometrically realistic with 36 joints and 88 degrees of freedom (not counting the hands). Each body motion has a body part whose movement is its primary purpose. Such a body part is called the *end effector* of the motion. At the level of gross motion, a hand, a foot, or the head is a typical end effector. These end effectors and the pelvis are called the *primary control parts*, because they are used to perform most important motions, e.g., viewing, reaching, bending, squatting, stepping, and walking. The other important control parts are elbows, shoulders, and eyes. The placement, i.e., the position and orientation of a control body part is specified in terms of the position of a point and the orientation of a vector associated with the body part. Such points and vectors are called the *control points* and the *control vectors* respectively. The motion of a control part is dependent on the base joint of the body linkage that is fixed while the control part is moving. That base joint is called the *pivot joint*. When we say that a control part is moved *with respect to* a joint, or a position is reachable by a control part *with respect to* a joint, that joint referred to is the pivot joint. The shoulder, the pelvis, or a foot is used as the pivot joint of hand motion, and a foot is used as the pivot joint of the pelvis motion.

Motions of body parts are classified into translational motions and orientational motions. The former are defined in terms of pivot joints, control points, and goal points. The latter are defined in terms of pivot joints, control vectors, and goal vectors (Table 1 and 2). For instance, an orientational motion, *happen(orient(Agent, Toe, PelvisVector, GoalVector), [T1,T2])* is a

6

motion in which the agent *Agent* rotates the pelvis with respect to the pivot joint *Toe* from time *T1* to time *T2* so that the control vector from the pelvis to the head, *PelvisVector*, is aligned with a given goal vector *GoalVector*. In addition, we have a primitive motion *walk*. The agent can walk following a curved sequence of foot steps.

Table 1: Translational Primitive Motions

Primitives	Pivot Joint	Control Point
move pelvis	toes	pelvis
move hand	shoulder	palm
move right foot	left toes	right foot
move left foot	right toes	left foot
move head, eye	waist	head, eye

Table 2: Orientational Primitive Motions

Primitives	Pivot Joint	Control Vector
orient pelvis	toes	pelvis vector
orient hand	shoulder	palm vector
orient right foot	left foot	right foot vector
orient left foot	right foot	left foot vector
orient head, eye	waist	head/view vector

Some of these primitives were first implemented by Phillips (Phillips and Badler 1991). Previously orientational motion of a control part was defined by the rotation axis and the rotation angle. Hence it is not possible to directly say "move the foot to a given position and make the foot direction from the heel to the toes parallel to a given direction'. This is acceptable when motion of a control part is controlled by direct manipulation, because the user can see the rotation angle of the foot. But when used by the planning program, indirect goal-directed specification is often more convenient. In the new system, in addition to the direction specification, the orientational motion of a control part is also specified by the control vector and the goal vector with which the control vector is to be aligned.

2.2 General Motion Rules

The posture planner uses the following assumptions, general motion rules, and preference constraints for motions.

1. The agent is supported only by its feet; it does not use its hands or other body parts to support itself. Posture planning uses information only about objects that are within the view of the agent. The agent tries to look at a given target object as early as possible and maintains its gaze toward the target object as long as possible.

2. Motions should respect both collision avoidance constraints and the body balance constraint.

3. It is unstable in terms of body balance to lower or raise the upper body, or move the upper body parts, e.g., the torso and hands, with only a single foot on the ground. These motions are always performed with the two feet rooted on the ground.

4. With both feet on the ground, upper body parts can be moved in parallel with the vertical movement of the pelvis. For instance, a hand can be stretched to the ground by bending the upper body, while the pelvis is lowered.

5. Stepping is performed with a standing posture as much as possible, because stepping with nonstanding postures is difficult.

6. Any body part tends to be placed so that the whole body's posture is most comfortable, subject to given constraints. The notion of comfort is captured by imposing a priority over primitive motions, so that motions considered more uncomfortable (by commensensical observation) are applied only when motions considered more comfortable do not work. For example, to reach an object, the agent tends to stretch his arm as much as possible while bending the pelvis as little as possible. Also, when stretching a hand to reach the ground from the standing posture, the agent bends the upper body at the pelvis rather than lowering the pelvis (by bending the knee), if bending the upper body works. If bending the knee is needed as well as bending the upper body, bending the knee is minimized. But in general, the ordering of comfortable motions depends on the nature of the task to be performed. For instance, the posture for lifting a weight should be different from the posture for simply touching it.

2.3 Specifying Planning Rules

The planning process is partly determined by the action formalism that defines planning rules, i.e., motion rules. A planning rule is represented by Definite Horn clauses as in *Prolog* (Kowalski 1976, 1986), whose head (consequent) is a goal and whose body (antecedent) can be a mixture of queries, temporal ordering constraints, physical constraints, subgoals, and primitive actions. Queries are used to ask for information about the world state and structure. Queries are frequently used to specify conditions under which the rule is relevant and to bind variables that satisfy the queries. The temporal ordering constraints are equality or inequality relations between time points. They constrain temporal ordering among conditions such as qualifiers, physical constraints, subgoals, and actions. Goals are conditions that the agent *intends* to achieve. Physical constraints are conditions which the agent *must* satisfy before or while given goals are achieved. Examples of them are collision avoidance constraints and the body balance constraint. A planning rule is interpreted as saying that (1) the rule is used only when the queries can be true, and (2) the head will be true if the subgoals and the actions are achieved while respecting the physical constraints and temporal ordering constraints.

For example, Table 3 and 4 show specification of two rules **Rule 1** and **Rule 2**. Spatial relations used in the rules are defined in Table 5.

Table 3: Rule 1

(a) holdat(within-view-of(GoalPos, Ag.eyes), T1),

(b) ?not holdat(within-comfortable-reach(GoalPos, Ag.Hand), T1),

(c) achieve(Ag, within-horizontal-reach(GoalPos, Ag.Hand), [T1, T2])

(d) achieve(Ag, within-comfortable-reach(GoalPos, Ag.Hand, Pivot), [T1,T2]),

(e) achieve(Ag, maximum(parallel(forward-axis-of(Ag.pelvis),

                     direction-of(ground-of(Ag.pelvis), ground-of(GoalPos)))), [T1,T2])

(f) achieve(Ag, collision-free(chain-between(Pivot, Ag.Hand)), [T2,T3])

(g) happen(move(Ag, Pivot, Ag.Hand, GoalPos), [T2,T3]),

$\Rightarrow$ (h) achieve(Ag, positioned-at(Ag.Hand, GoalPos), [T1,T3]).

    Rule 1 defines a goal of moving a hand to goal position *GoalPos* from time *T1* to *T3*, with respect to pivot joint *Pivot*. The conditions (a) - (i) of the rule (1) above are conditions used to achieve the goal of the moving of the hand to the goal position. Formulas of the form ?G are *queries*. The meaning of each condition is as follows.

(a) At time *T1*, goal position *GoalPos* should be within the view of the agent.

(b) At time *T1*, the goal position is *not* within the comfortable reach of the hand.

(c) From time *T1* to time *T2*, the agent should achieve the goal of placing the body so that the goal position is within the horizontal reach of the hand.

(d) From time *T1* to *T2*, the agent should place the body so that the goal position is within the comfortable reach of the hand.

(e) From time *T1* to *T2*, the agent should make the forward axis from the pelvis maximally parallel to the ground projection of the direction from the pelvis to the hand goal position subject to other constraints.

(f) From time *T2* to *T3*, the agent should move the hand to the goal position with respect to *some* pivot joint *Pivot*.

(g) The body chain between the chosen pivot joint and the hand should be collision free throughout the motion from time *T2* to *T3*.

Table 4: Rule 2.

(a) ?position-of(PelPos, Ag.pelvis, within-horizontal-reach(GoalPos, Ag.Hand)),

(b) achieve(Ag, positioned-at(Ag.pelvis, PelPos), [T1,T2]),

$\Rightarrow$ (c) achieve(Ag, within-horizontal-reach(Ag.Hand, GoalPos), [T1,T2]).

    Rule 2 defines a subgoal of **Rule 1**. It means: To achieve goal (c) *within-horizontal-reach(Ag.Hand, GoalPos)* during interval *[T1,T2]*, (a) a position of the pelvis *PelPos* such that *within-horizontal-reach(GoalPos, Ag.Hand)* should be first found and then (b) the agent should move the pelvis to that position

Table 5 : Definition of Spatial Constraints

- The *frontal orientation* constraint: Let the space be divided into the front half-space and the back half-space by the frontal plane, i.e., the vertical plane passing through the center of the body side to side. Suppose that the goal position of the end effector is in the back half-space of the body. If the goal can be achieved simply by orienting the body (with or without stepping), the agent does it. Otherwise, the agent prefers to have the goal position of the end effector in front of the body, that is, in the frontal half-space with respect to the body.

- The *horizontal reach* constraint: The goal position of the end effector should be within its *maximal horizontal reach*. When a hand is the end effector, the length of an arm is used as the maximal horizontal reach of the end effector. The agent can maintain the horizontal reach of the hand to be equal to the length of the arm whether the agent moves the hand (1) while standing, (2) while bending the upper body at the pelvis without squatting, i.e., without bending the knee, or (3) while squatting with or without bending the upper body. Suppose that the goal position of the end effector is somewhere under a big table. The horizontal reach constraint is used to determine which side of the table and which part of that side to approach to reach the goal position. When the end effector is the head, the maximal horizontal reach of the end effector is considered the length of the upper body from the pelvis to the head.

- The *comfortable reach* constraint: It says that the goal position should be within a *comfortably reachable region* of the end effector. The comfortably reachable regions of a hand are given as properties of the agent. The global placement of the reachable region of the hand is a function of the four control parameters: (i) the ground position of the body, (ii) the horizontal orientation of the forward pointing vector of the body, (iii) the vertical position of the pelvis (the body center), and (iv) the orientation of the torso vector, the vector from the pelvis to the head.

3 Planning and Execution

Here we describe an overall structure of the planning process in which symbolic reasoning and algorithmic problem solving is integrated. The overall control of planning and execution is depicted in figure 3. It has four subprocesses (indicated by boxes): **goal reduction, constraint solving, collision discovering,** and **execution.** Section 3.1 describes the overall flow of planning and execution. Section 3.2 and 3.3 describe the four subprocesses and a strategy for avoiding the local minimum problem.

3.1 The Overall Flow of Planning and Execution

We summarize the overall flow of planning and execution. Table 6 describes a rigorous control flow of the planning algorithm.

The input goal typically specifies only the goal of the end effector. Hence, to initiate posture planning, the agent must determine the goal positions and orientations of important body parts, e.g., the feet, the pelvis, and the head which would enable the end effector goal to be achieved. To do so, the **goal reduction** process finds a planning rule whose head can be identical with the input goal and reduces the goal into the conditions on the body part of the rule, if the queries are true with respect to the current simulated world at the moment of reduction. The generated conditions are added to the current plan. The current plan is represented as a goal tree in which children nodes are subgoals, constraints, or primitive
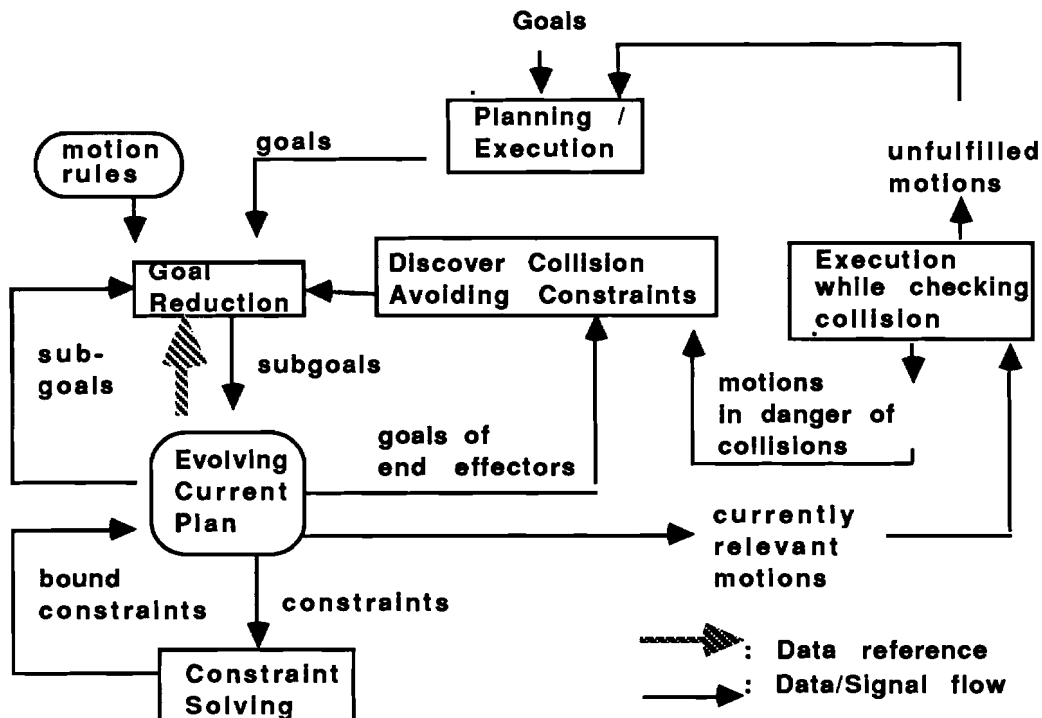
Goals

Planning /
Execution

motion
rules

goals

unfulfilled
motions

Goal
Reduction

Discover Collision
Avoiding Constraints

Execution
while checking
collision

sub-
goals

subgoals

Evolving
Current
Plan

goals of
end effectors

motions
in danger of
collisions

currently
relevant
motions

bound
constraints

constraints

Constraint
Solving

: Data reference
: Data/Signal flow

Figure 3: The overall control of planning and execution
invokes the four subprocesses (boxes) at appropriate times.

motions for the goals of parent nodes. The current plan is also used as an assumption by
the **goal reduction** process when it checks if a given condition is true with respect to the
current state of planning. Goals of the current plan are further reduced. Collision avoidance
constraints are reduced to their subgoals if the reduction rules are provided. Otherwise, the
collision avoiding constraints are discovered. For instance, the **collision discovering** process
generates collision avoiding constraints of the end effector at planning time, because the motion
of the end effector can be predicted at planning time. When goals are reduced to subgoals of
finding the positions and orientations of the feet, the pelvis, the head, and the hand satisfying
their spatial constraints, the **constraint solving** process is invoked. For instance, suppose
the following rule is used for goal reduction

(a) ?position-of(PelPos, Ag.pelvis, within-horizontal-reach(GoalPos, Ag.Hand)),

(b) achieve(Ag, positioned-at(Ag.pelvis, PelPos), [T1,T2]),

⇒ (c) achieve(Ag, within-horizontal-reach(Ag.Hand, GoalPos), [T1,T2]).

The planner should find values of variable *PelPos* that satisfies clauses (a) and (b). Since
variable *PelPos* is real-valued, it needs special treatment. The planner delays solving con-
straints of real variables constraining the placement of the body until other constraints con-
straining the body placement, e.g. the orientation of the torso upward vector, are generated
by other goal reductions, so that they would be solved simultaneously (see the **constraint**

11

solving part of the planning algorithm in Table 6).

Table 6: The Planning Algorithm

The planner maintains several state variables. The queue of the current goals *CurrGoals* is initialized to the input set of goals. The planner maintains the current plan *CurrPlan* and the current temporal constraints *Protection*, a set of protected intervals in which achieved goal conditions should be protected. It maintains a list of constraints *Constraints*, a set of conditions for which there are no goal reduction rules. The variables occurring in *Constraints* may be bound in the process of goal reduction. The planner also maintains the current (simulated) world *World* which is initialized to the initial world. The planning process **Plan** is a recursive procedure.

Plan*(CurrGoals, Protection, CurrPlan, Constraints)* :

1. If *CurrGoals* is empty, return with success; Otherwise, select a goal G from the goal queue *CurrGoals*.

2.1. **Querying**: If G is a *query* of the form *?G0* then try to find if it is true with respect to the current world *World*. If true then delete G from *CurrGoals* and call *Plan* recursively. Otherwise, backtrack to the previous decision point, where the planner will try an alternative decision.

2.2. If G is a temporal ordering goal $M < T0$ or $M > T1$, add it to *Constraints*. Call *Plan*.

2.3. **Goal Reduction** (base case) : If G is a *goal* and is unifiable with a condition F in the current world *World*, with variables of G being bound, then delete G from *CurrGoals* and propagate the binding to the new *CurrGoals* and *Constraints*. Add the condition F to *Protection*. Call *Plan*.

2.4 **Action Encountered** : If G is an action of the form *happen(ActType, [T1, T2])*, do the following: If time parameter *T1* or *T2* is unbound, either bind them to the times of an old action of the same type *ActType* or bind them to new time constants. Add the bound action B to *CurrPlan*. Call *Check-Consistency-of-Plan(World, Protection, CurrPlan, Constraints, B)* to see if the effects of the new action B are compatible with the conditions of *Protection* or physical constraints. If consistent, call *Plan*. Otherwise, backtrack.

3. **Goal Reduction** (recursive) : If G is unifiable with consequent A of a planning rule $D \Rightarrow A$, with variables of G being bound, propagate the binding to D, *CurrGoals* and *Constraints*. Delete G from *CurrGoals*, add the bound G to *CurrPlan*, and add the bound D to *CurrGoals*. Call *Plan*.

4. **Constraint Solving** : If there are no goal reduction rules for goal G, add it to constraint list *Constraints*. (Solving it may be postponed so that the goal reduction process may add a sufficient number of constraints to be satisfied simultaneously.) Solve *Constraint* if desirable. Call *Plan*.

Check-Consistency-of-Plan*(World, Protection, CurrPlan, Constraints, B)*:

1. Derive the unconditional and conditional effects of action B with respect to *World*, *Protection*, *CurrPlan*, and *Constraints* using the planning rules and deduction rules in the forward chaining manner, and add them to *World*.

2. For each protected condition in *Protection*, see if it still holds with respect to the new world *World*. If not see if it can be made to hold by imposing additional temporal constraints on *CurrPlan*. If that is not possible, remove it from *Protection*. It means that achievement of a goal condition is retracted.

12

To find a solution of a constraint set, a domain-specific discrete and rule-driven search is used. At a given moment, the motions of the current plan are executed by the **execution** process, if they are currently relevant. The motions being executed may be in danger of collision because collision avoidance planning at planning time is not complete, or may not be fulfilled for some reasons. In each case a new cycle of planning is invoked (see the data flows going out from the **execution** process in figure 3). While the collision avoiding constraint of a body part is achieved, the primary goal of the end effector, e.g. a hand, may be temporarily disabled to avoid the competition with the secondary but more urgent goal of collision avoiding. The tight interleaving of planning and execution is based on the fact that by nature posture planning is approximate and thus feedback from execution is necessary. This interleaving framework belongs to the recent planning paradigm called *reactive planning* (Schoppers 1987, 89; Ambros-Ingerson 1988), which takes the perceiving-acting cycle seriously.

3.2 Subprocesses

The subprocesses will be explained using a goal "pick up the block under the table" in figure 1.

If the target object is not seen by the agent, the **goal reduction** process suggests a *viewing goal* so that the target object will be seen. For example, if the block is too far under the table and is not within the view of the agent, the **goal reduction** process will generate a viewing goal of placing the head of the agent in front of an edge of the table and below the height of the table top so that the agent can see the block under the table. If the target object of a given goal is seen by the agent at the initial situation, the goal position of the end effector, e.g., a hand, is determined from the nature of the input goal. In the current example (figure 1), the target object, the block, is seen by the agent at the current situation.

Collision discovering process generates collision avoiding constraints of the end effector at planning time, because the motion of the end effector can be predicted at planning time. For the goal of picking up the block under the table, collision avoiding constraints of the hand are generated such that the hand at the goal position should be under the table top and horizontally away from the left and right legs of the table.

Now the goal position and collision avoiding constraints of the end effector are given. To find the positions and orientations of the important body parts at the goal region, the **goal reduction** process generates constraints of the body placement whose satisfaction would lead to achievement of the end effector goal. More precisely, the **goal reduction** process generates goal conditions and constraints about the body placement such that (1) it satisfies the *minimum walking* constraint and (2) the *frontal orientation* constraint; and the goal position of the end effector satisfies (3) the *horizontal reach* constraint and (4) the *comfortable reach* constraint.

To find the body placement, that is, the positions and orientations of the feet, the pelvis, the head, and the hand satisfying the four body placement constraints, the **constraint solving** process is invoked (figure 3). The **constraint solver** is an iterative process that suggests how to change the control parameters to achieve given constraints while achieving the end effector goal. To search for values of the control parameters that will satisfy given constraints, the

constraint solver suggests discrete values of the control parameters, determines the posture satisfying the positions and orientations denoted by those values, and then checks if the posture satisfies given constraints. A discrete search is used instead of a continuous search, e.g., search based on the gradient of some objective function representing a given motion goal, because the planning stage spatial reasoning is approximate at best. The **constraint solver** uses a dependency relation among the control parameters that indicates which control parameter should be bound first. For instance, the foot positions should be bound before all the other control parameters are determined. The posture satisfying the instantiated positions and orientations of the important body parts is found by an inverse-kinematics body positioner (Zhao 1989). The body positioning algorithm uses the joint space formulation typically used in robot motion planning. But note that it is not used for *planning*, but only for determining exact postures corresponding to the planned values of the control parameters and thereby for checking the feasibility of the suggested plan all of whose variables are instantiated. In a cluttered workspace, the suggested plan may often be infeasible and the posture planner may not be able to suggest a workable plan, because it uses only task space measurements to suggest a motion plan. In such a situation, we can use a full-fledged robot motion planner, e.g., a path planner (Ching 1991) that finds a sequence of postures that satisfy joint strength constraints as well as collision avoidance constraints. But note that dropping down to the joint space formulation occurs only when further planning using the task space measurements is no longer feasible.

The four constraints on the placement of the body at the goal region are concerned only with the current end effector, a hand. Thus the posture satisfying the four constraints does not guarantee collision-free motion of body parts other than the end effector especially the upper body including the head. The **constraint solver** must consider other body parts when solving the four constraints. Suppose that the posture determined by the initial values of the task space control parameters would cause the head to collide with a face of the table (confer figure 4). The **constraint solver** suggests that the head should be positioned so that its spatial relation with respect to the table would be the same as the spatial relation of the end effector with respect to the table, that is, in front of the table. This constraint on the head position is based on the fact that it should not hinder achieving the goal of the hand. The placement of the head depends on (i) the orientation of the body, (ii) the orientation of the torso upward vector, and (iii) the height of the pelvis. In the current example, the **constraint solver** tries to keep the height of the body and the horizontal position of the body center of the initially suggested final posture while allowing the orientation of the body about the vertical axis to be changed (see figure 5). In other words, the *horizontal reach* constraint and the *comfortable reach* constraint are considered more important than is the *frontal orientation* constraint. Then, the **constraint solver** determines the height of the pelvis and the orientation of the torso upward vector that allows the end effector to reach the block (figure 6). (The enclosed video tape shows the rule-driven constraint solving process by which the goal posture is determined.)
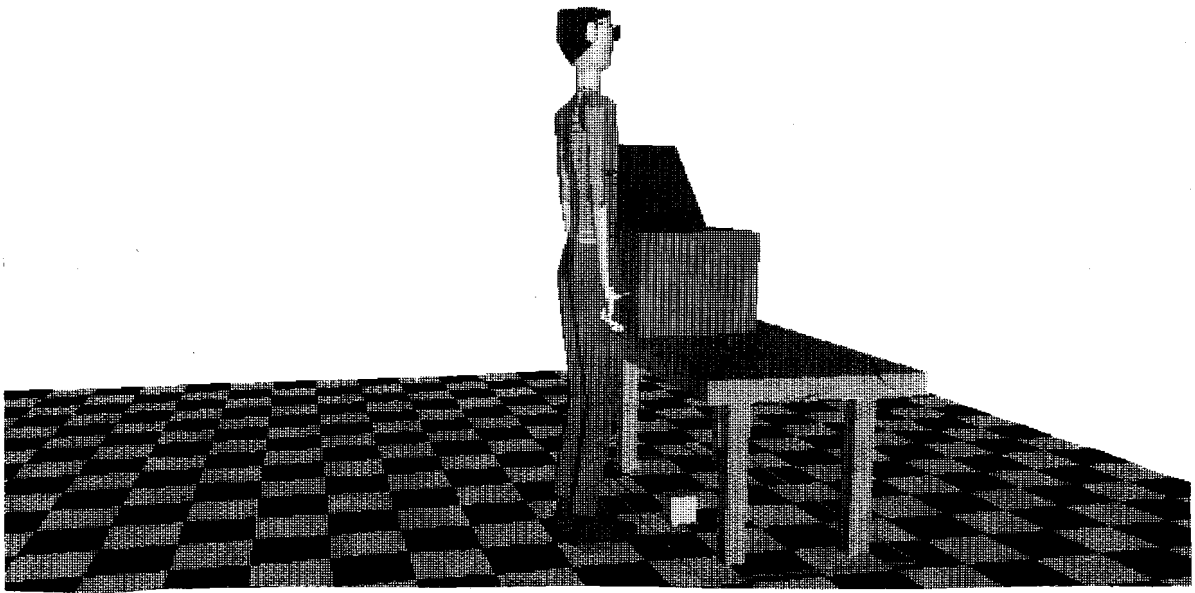
Figure 4: A goal posture that would cause the upper body to collide with the table top.
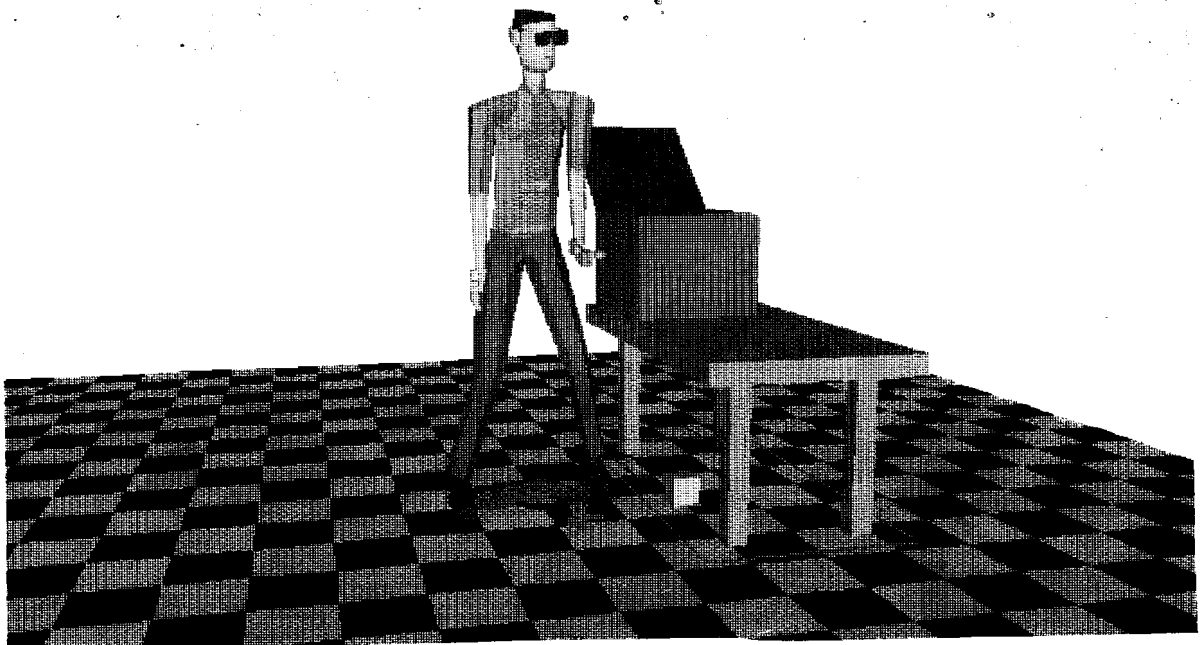
Figure 5: Orienting the body to avoid collision of the head.

15
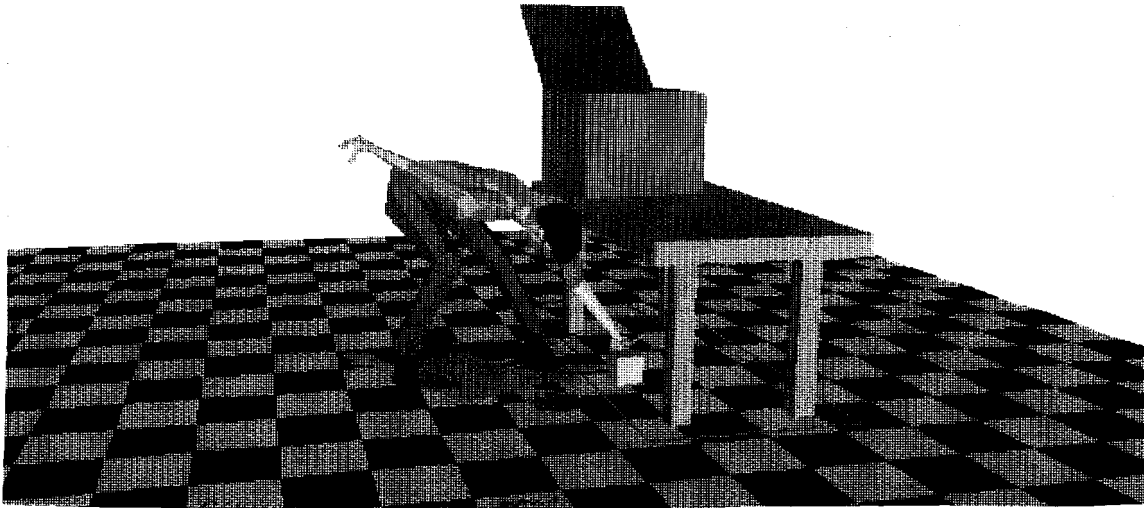
Figure 6: Squatting and bending to reach the block.

When the final goal posture is determined in terms of the positions and orientations of the feet, the pelvis, the head, and the hand, the **goal reduction** process generates, using the motion rules, a sequence of primitive motions, e.g., stepping, orienting the body, bending with respect to the pelvis or the feet, or squatting, which leads to the final goal posture.

3.3   Avoiding the Local Minimum Problem

In the task space formulation of motion planning, the local minimum problem is frequently encountered. Here we review a typical task space formulation (Khatib 1986, Barraquand et al. 1989) and see how the local minimum problem is attacked. Because the motion planning process can be explained using the metaphor of *potential energy*, the method is called *potential field* approach. The motion of the body is described by motion of control points on the body. The control points have both an attracting potential function due to their goal positions and a repelling potential function due to obstacles. The attracting potential energy of a control point is represented by a function whose magnitude is proportional to the task space distance from the control point to its goal position. The repelling potential energy of a control point is represented by a function whose magnitude is inversely proportional to the task space distance from the control point to the obstacle surfaces. The potential energy function of the whole moving body is obtained by combining the attracting and repelling potential energy of the individual control points. The body is incrementally moved in the direction that minimizes this combined potential energy. The intermediate configurations of the moving body constitute the motion of the body.

The potential energy of the control points, more precisely, the force vectors induced from the potential energy may compete with each other and cancel each other, an attracting force against another attracting force and a repelling force against an attracting force. Therefore,

16

the combined potential energy may be minimized even when the individual potential energies of the control points are not. To alleviate the local minima problem, Barraquand et al. (1989) suggested different ways of combining potential functions of control points based on the task space consideration. For example, the potential energy of the control point that is the farthest away from its goal position in terms of the task space distance is given the greatest weight and all the other potential energies the *zero* weight. Using the task space criteria to avoid the local minima problem may be useful in some cases. Phillips and Badler (1991) also used a weighting scheme to deal with the local minimum problem caused by multiple simultaneous goals. However, weighting schemes using task space criteria are arbitrary in nature and try to solve the problem without looking at the causes. Thus we do not know exactly how weights given to task space goals influence the joint angles of the body, which ultimately determine the posture.

It seems that solving the local minimum problem requires reasoning with respect to the joint space. It is illustrated using examples. In our human figure motion system, the figure is modeled as a branched joint chain typically rooted at one of the feet. Suppose the figure is rooted at the **left** foot. The placement of the human figure is maintained by continuous satisfaction of spatial constraints on important body parts. Both feet are to lie on the ground plane and the center of mass of the body is to lie on a vertical line passing through the body to maintain balance. Bending the pelvis forwards conflicts with moving the pelvis down. Bending the pelvis requires the knee joint to be extended, while the lowering the pelvis requires the knee joint to be bended. This fact can be predicted by comparing the gradient of the potential function P_1 for bending the pelvis with the gradient of the potential function P_2 for lowering the pelvis. The (negative) gradient of a potential function (with respect to the joint space) can be viewed as a *force* that "moves" points on the graph of the potential function in the fastest descending direction. This force of a potential function has as many components as the number of the joint variables. Let the force F_1 of potential function P_1 be the *bending force*. Let the force F_2 of potential function P_2 be the *lowering force*. Both forces have three components, the first for the ankle joint, the second for the *knee joint*, and the third for the hip joint. The knee component of the bending force F_1 has the opposite sign of the knee component of the lowering force F_2. It means that bending the pelvis and lowering the pelvis give opposite orders to the knee joint. To avoid this situation, the posture planner adjusts the gradients, that is, the bending force and the lowering force so that both forces do not conflict at any component and the magnitudes of both remain to be the same as before. In the case of the current example, it is achieved by the following reasoning:

1. Both the knee joint and the hip joint are major contributors to bending the pelvis, while the only the knee joint is a major contributor to lowering the pelvis.

2. Hence, let the knee joint contribute to lowering the pelvis, because bending the pelvis has another major contributor, the hip joint.

3. If only the hip joint contributes to bending the pelvis, the force for bending the pelvis becomes weaker than the original bending force F_1 due to both the hip joint and the knee joint. Hence to preserve the magnitude of the force, increase the hip joint component of the bending force F_1 when producing the new bending force.

The above strategy is general so that it can deal with situations difficult to foresee. For example, bending the pelvis forward conflicts with keeping the right (nonroot) foot on the ground. Bending the pelvis forward requires bending the left hip joint, which causes the right foot move backward and off the ground. The above strategy will make the left hip joint contribute only to bending the pelvis, because the body can use the right hip joint and the other joints below it to make the right foot positioned on the ground.

The posture planner does not use the repelling potential functions to avoid collision because it is difficult to predict and control their influences and their competition with the attracting potential functions. Rather, a new cycle of planning is invoked to suggest candidate goal positions and orientations of the body parts in danger of collision. Also, to avoid competition between achieving the goal and avoiding collision, the posture planner disables the process of achieving the goal temporarily while the secondary goal of avoiding collision is achieved.

4 Conclusion

We developed a human motion planning method which uses both symbolic planning techniques and algorithmic problem solving techniques. In this method, motion planning comes down to characterizing intermediate postures of the body in terms of the task space positions and orientations of the control points and vectors associated with the feet, the pelvis, the head (eyes), and the hand. To do so, human motions are axiomatized at a gross level and intermediate postures are roughly determined via goal reduction reasoning by means of the motion axioms (rules) and a procedure to discover specific collision-avoidance constraints. The present study shows that human motion and spatial reasoning can be captured by the symbolic reasoning method to a considerable extent, while an algorithmic approach is also needed to take care of geometric details. For instance, certain motion rules, e.g., that orienting the body about the vertical axis requires stepping when the angle goes beyond a given threshold, can be formulated symbolically with respect to the *task space*. The joint space formulation is used only when it is absolutely needed. That is, the joint space formulation is used by an algorithm only for checking if task space conditions of given control points and vectors of the body are satisfiable or physically feasible with respect to the current simulated world. Also, the *rule-based* symbolic reasoning gives the ability to use various motion rules and constraints based on human factors analysis, say, for particular agents or groups of agents. For example, given agents who have weak waists, the designer can write down motion rules so that bending motion is avoided as much as possible. Then, the user can figure out if the design of the workspace is suitable for such agents via task simulation. This is an advantage over the animation system with the hard-coded control structure.

Finally, the present study has a more general significance as a case study of integrating the AI symbolic/discrete reasoning paradigm and numeric/continuous mathematical modeling of applied mathematics. We believe that the integration of classical methods of continuous mathematics into a knowledge-based reasoning system is worthy to pursue. The mathematical methods are useful to deal with the continuous domain, to which the task space of agents belongs. The symbolic reasoning method helps organize information and control needed to solve a given problem through more explicit representation of assumptions and high level

Fikes, R.E. and N.J. Nilson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2(1971).

Jung, M. R., J. Kalita, N. Badler, and W. Ching. Simulating Human Tasks Using Simple Natural Language Instructions. The *Winter Simulation Conference Proceedings*, December 1991, Phoenix, Arizona.

Khatib, O. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots, *The International Journal of Robotics Research*, Vol. 5. No. 1, Spring, 1986. pp.90 - 98.

Kowalski, R. Logic for Problem Solving. Elsevier Science Publishing Co.:New York. 1976.

Kowalski, R. and M. Sergot. A logic-based Calculus of Events, *New Generation Computing*, 4(1986) 67-95.

Lee, Phillip, S. Wei, J. Zhao, & N. Badler. Strength Guided Motion. *Computer Graphics*, Vol. 24, No. 4, pp. 253-262. 1990.

Lozano-Perez, T. Spatial Planning: A Configuration Space Approach. *IEEE Transactions on Computers*. Vol C-32. No. 2. (1983).

Lozano-Perez, T. Task Planning in: Brady M (ed.) *Robot Motion: Planning and Control.* MIT press, Cambridge, MAss, 1985.

Lozano-Perez, T. A Simple Motion-Planning Algorithm for General Robot Manipulators. *IEEE Journal of Robotics and Automation*, Vol RA-3, No. 3, June 1987.

Maciejewski, A. and C. A. Klein. Obstacle Avoidance of Kinematically Redundant Manipulators in Dynamically Varying Environments, *The International Journal of Robotics Research*, Vol. 4. No. 3, Fall 1985. pp. 109-117.

Monheit, Gary, & Norman I. Badler. *A Kinematic Model of the Human Spine and Torso.* IEEE Computer Graphics and Applications 11(2), March 1991.

Munson, J. H. 1971. Robot planning, execution, and monitoring in an uncertain environment. *Proceedings of the 2nd IJCAI, 1971.*

Phillips, C., J. Zhao, & N. I. Badler. Interactive Real-time Articulated Figure Manipulation Using Multiple Kinematic Constraints. *ACM Computer Graphics 24,2 (1990).* pp. 245-250.

Phillips, Cary & Norman I. Badler. Interactive Behaviors for Bipedal Articulated Figures.

Reynolds, C. W. Flocks, herds, and schools: a distributed behavioral model. *Proc. SIGGRAPH '87, Computer Graphics*, 21, (4), 25-34. 1987.

Renault, O., N. M. Thalmann and D. Thalmann. A Vision-based Approach to Behavioral Animation. *Visualization and Computer Animation*, Vol. 1. No. 1, August 1990.

Ridsdale, G., S. Hewitt, and T.W. Calvert. The Interactive Specification of Human Animation. *Proc. Graphics Interface '86*, May, 1986, Vancouver, Canada.

Sacerdoti, E.D. *A Structure for Plans and Behavior.* New York: Elsevier North-Holland, Inc. 1977.

Schoppers, M. Universal plans for reactive robots in unpredictable domains. In *Proceedings of the 10th IJCAI, 1987.*

Schoppers, M. *Representation and Automatic Synthesis of Reaction Plans.* PhD thesis, University of Illinois at Urbana-Champaign, Illinois, 1989.

Tate, A. Interacting Goals and their Use. In *IJCAI-75.* 1975.

Tate, A. Generating Project Networks. In *IJCAI-77.* 1977.

Waldinger, R. Achieving Several Goals Simultaneously. In Elcock, E. and Michie, D, ed., *Machine Intelligence 8*, Ellis Horwood, Chichester, England, 1977, pp. 94ff.

Wilkins, D.E. Domain-independent Planning: Representation and Plan Generation. *Artificial Intelligence 22(1984)*, 269ff.

Witkin, A. and M. Kass. Spacetime Constraints. *Computer Graphics*, 22(4):159-168, 1988.

Zeltzer, D. Towards an integrated view of 3-D computer animation. *The Visual Computer*, 1(4):249-259, 1985.

Zhao, J., & N. I. Badler. *Real Time Inverse Kinematics with Spatial Constraints and Joint Limits.* Technical Report MS-CIS-89-09, Computer and Information Science, University of Pennsylvania, PA, 1989.