Synthesizing Animations of Human Manipulation Tasks

Katsu Yamane*
The University of Tokyo

James J. Kuffner†
Carnegie Mellon University

Jessica K. Hodgins‡
Carnegie Mellon University

Abstract

Even such simple tasks as placing a box on a shelf are difficult to animate, because the animator must carefully position the character to satisfy geometric and balance constraints while creating motion to perform the task with a natural-looking style. In this paper, we explore an approach for animating characters manipulating objects that combines the power of path planning with the domain knowledge inherent in data-driven, constraint-based inverse kinematics. A path planner is used to find a motion for the object such that the corresponding poses of the character satisfy geometric, kinematic, and posture constraints. The inverse kinematics computation of the character's pose resolves redundancy by biasing the solution toward natural-looking poses extracted from a database of captured motions. Having this database greatly helps to increase the quality of the output motion. The computed path is converted to a motion trajectory using a model of the velocity profile. We demonstrate the effectiveness of the algorithm by generating animations across a wide range of scenarios that cover variations in the geometric, kinematic, and dynamic models of the character, the manipulated object, and obstacles in the scene.

CR Categories: 1.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: human figure animation, motion capture, path planning, two-hand manipulation

1 Introduction

Human figures are difficult to animate because of the need for choreographing many degrees of freedom so that they will move in a coordinated and a humanlike fashion. Two classes of semi-automatic techniques have been developed for creating human animations: *model-based* and *data-driven*. Model-based approaches use simulation, search, and optimization to generate character motion by restricting the space of possible motions via kinematic, dynamic, or biomechanical models. This approach provides a flexible and compact representation of motion, but can sometimes be difficult to construct and control, or may fail to generate natural-looking motions if the models do not sufficiently constrain the motion. Data-driven approaches utilize captured motion to create animations that contain the subtle movement details recorded from a human actor. These do not provide a perfect solution either, however, because adapting existing captured motions to new situations is not easy, particularly when the motion is highly dynamic or involves constrained interactions with the environment or other characters. We focus on such constrained interactions in this paper.

*e-mail: yamane@ynl.t.u-tokyo.ac.jp
†e-mail: kuffner@cs.cmu.edu
‡e-mail: jkh@cs.cmu.edu

Figure 1: Synthesized animations of manipulation tasks.

We present a technique for synthesizing motion for manipulation tasks that attempts to combine the good features of model-based and data-driven approaches. Given user-specified start and goal positions for an object, our algorithm relies on a randomized planning algorithm to find a feasible path for the manipulated object. The planning process is informed by a database of natural human postures for similar tasks and model-based balance and collision constraints. For each position and orientation of the object, an iterative inverse kinematics method calculates a pose for the character that satisfies posture and environment constraints. In order to bias the search toward natural-looking poses, we use a database of captured motion to derive additional soft constraints that influence the character's pose. After a path is found, it is locally optimized for smoothness and parameterized in time based on a velocity profile model derived from human motion data. Because gaze direction is not part of our motion capture database, we use a model of gaze direction to automatically coordinate the character's head and eye movements with the motion of the rest of the body.

Our algorithm synthesizes natural-looking, whole-body motion for a wide range of manipulation tasks while respecting environment and posture constraints. We demonstrate the power of this approach with examples such as those shown in Figure 1. The algorithm is robust to variations in the geometric, kinematic, and dynamic models of the character, the manipulated object, and obstacles in the scene. We do not address the problem of selecting or planning a grasp and assume that the character is able to manipulate the object with simple pre-defined grasp locations.

2 Background

Methods for generating and modeling human motion have been developed and studied widely in biomechanics, robotics, and computer graphics. Most human motions are severely underconstrained

because there are many ways to accomplish a task. Model-based algorithms address this problem by relying on a model of human motion to constrain the search space sufficiently that manual or automatic techniques can be used to find a natural-looking solution. Many different kinds of models have been used over the past fifteen years including dynamics, kinematics, and more task-specific models such as minimum torque change for reaching motions [Kawato 1999] or pelvis swing during walking [Bruderlin and Calvert 1989]. While each of these models captures some of the salient aspects of human motion, none of them captures a wide range of activities at a sufficient level of detail to match the quality of a specially captured sequence.

Playing back captured human motion data on an animated figure is an excellent way to reproduce the naturalness, style, and subtle details of human movement; however, adapting data to new characters or situations is often difficult. Inverse kinematics, dynamic constraints, and optimization have been used to blend motion capture sequences [Rose et al. 1996] or to edit motion [Gleicher 1998; Gleicher 1997]. Like the approach described in this paper, these techniques bridge between data-driven and model-based algorithms. More recently researchers have explored algorithms that rely on having significant amounts of data and using it to build databases from which new motions can be assembled [Lee et al. 2002a; Kovar et al. 2002; Arikan and Forsyth 2002; Arikan et al. 2003]. For generating human arm reaching motion, Park and Chaffin interpolate from among multiple sets of motion capture data [Wiley and Hahn 1997; Park et al. 2002]. The idea of including certain poses as a seed for inverse kinematics was also explored by Rose, Sloan, and Cohen [Rose et al. 2001] in a system that built on their previous work [Rose et al. 1998]. Their approach differs from the one that we have taken in that the artist selected relatively few frames while the model was provided by a radial basis interpolation of the motion capture data. In our approach, the entire database is used in the inverse kinematics computation while the model is provided by the task constraints of manipulation, balance, and collision avoidance.

Motion planning techniques calculate collision-free trajectories in the presence of arbitrarily placed obstacles by searching the system configuration space for a collision-free path connecting a start configuration to a goal configuration [Latombe 1991]. Although reasonable performance can be achieved for low degree of freedom problems (low dimensional spaces), optimal motion planning algorithms based on exhaustive search are impractical for searching spaces with a large number of degrees of freedom. Heuristic motion planning algorithms were designed to solve problems in higher dimensions (e.g., [Barraquand and Latombe 1990; Kavraki et al. 1996; Liu and Badler 2003]). These planning algorithms have been successfully applied to the automatic animation of grasping and manipulating an object with a rooted torso [Koga et al. 1994; Bandi 1998; Kuffner 1999], locomotion with arm reaching [Kalisiak and van de Panne 2000; Pettre et al. 2003], and body posture interpolation [Ching and Badler 1992; Bindiganavale et al. 1994; Jung et al. 1994; Boulic et al. 1997].

As with other model-based techniques, planning can sometimes fail to generate natural-looking motion if the model is not sufficiently restrictive. For this reason, our approach is to plan in the six dimensional space of the object motion but then to use the feasibility of the resulting whole-body motion in the evaluation of the plan. Simeon and colleagues [2002] also combine inverse kinematics and planning to build a probabilistic roadmap for a six degree-of-freedom arm manipulating with regrasping. They address a far more complicated planning problem than that addressed here; however, our system computes whole body motion for a human character with the goal of making the motion appear natural rather than merely accomplishing the task.

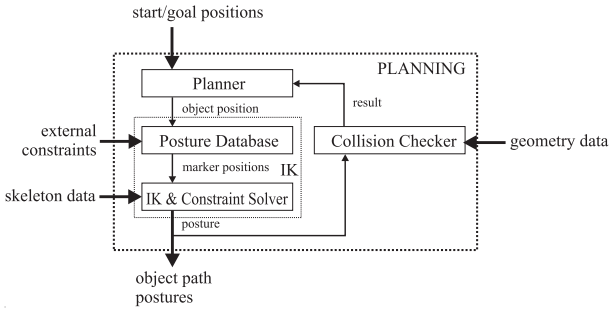


Figure 2: Overview of the planning phase.

3 Generating Motion for Manipulation

The motion synthesis strategy presented here attempts to combine the flexibility and generality of motion planning with the naturalness of captured data. We focus on automatically generating manipulation motions that allow human-like characters to reach an object and move it to a goal location and orientation. We present an overview of the algorithm and then explain each of the system components in detail.

The computation is divided into two phases: *planning* and *post-processing*. The planning phase generates a collision-free path for the object while taking into account the naturalness of the poses the character must use to position the object in a given location and the task constraints (balance and collision avoidance) (Figure 2). The post-processing phase then smooths the resulting path and applies a velocity profile based on human motion data. The user specifies the following information as input: 1) The start and goal locations of the object to be manipulated and the hand grasp constraints (the position and orientation of the hands relative to the object); 2) External constraints (constraining the feet to the floor, for example); 3) A character model, including skeleton kinematics and joint limits (used for inverse kinematics), geometry data (used for collision checking), and dynamic properties (used for balance constraints). The motion is planned in segments, with a new segment created each time that the contact specification changes. For example, a motion that involves contacting an object, moving it to a new location, and releasing it is planned as three separate motions with coincident start and end configurations. We treat the motion as separate segments because the constraints of the system and hence the planning problem differ when the character is contacting an object and when it is not. The captured posture database is similarly divided into segments based on contact conditions using a threshold on the velocity of the manipulated object.

3.1 Planning: Path Search

The planner searches the object space for a path that will move the object from the start configuration, q_{init}, to the goal configuration, q_{goal}, in the configuration space $\mathscr{C}$ of the manipulated object (a six-dimensional space of translations and rotations). The algorithm, a type of filtered RRT-Connect planning strategy [Kuffner et al. 2002], searches for a collision-free path connecting q_{init} and q_{goal} and lying entirely in $\mathscr{C}_{free}$, the open subset of collision-free configurations in $\mathscr{C}$. For a detailed description and analysis of Rapidly Exploring Random Trees (RRT) planning algorithms in a robotics context, the reader is referred to LaValle and Kuffner [2001]. RRT planning algorithms require a metric for the distance between the pair of configurations $q_1 = (x_1, R_1)$ and $q_2 = (x_2, R_2)$, where x and R denote the position and orientation components respectively. We

use a heuristic scalar metric for general 3D affine transformations:

$$\rho(q_1,q_2) = w_x||x_1 - x_2||^2 + w_R||rotdist(R_1,R_2)||^2 \qquad (1)$$

Unit quaternions are used to represent rotations, so the distance metric for the rotation component $rotdist(R_1,R_2)$ is a function that returns a normalized scalar number proportional to the great-circle arc distance between R_1 and R_2 on the 4D quaternion sphere. The translation and rotation components are weighted by the coefficients w_x and w_R.

Algorithm 1: PLANMANIPPATH(q_{init}, q_{goal})

```
T.init(q_init) // Initialize search tree;
while time < t_max do
    q_sample ← SELECTTARGET(q_goal, 0.05);
    result ← GROWTREE(T, q_sample);
    if result = REACHED and q_sample = q_goal then
        p ← EXTRACTPATH(T);
        SMOOTHPATH(p);
        return SUCCESS;
    end
end
return FAILED;
```

Algorithm 1 shows pseudocode for the overall planning algorithm. The core of the planning algorithm requires incrementally growing additional branches on a tree of connected free configurations. We begin by initializing the tree $\mathscr{T}$ with the single node q_{init}. The planning loop consists of repeatedly selecting a target sample configuration q_{sample} and attempting to grow the tree. SELECTTARGET() chooses the goal configuration q_{goal} as the sample configuration q_{sample} with some fixed probability (e.g. $P = 0.05$), and otherwise calculates a uniform random sample of the volume $\mathscr{V} \subset \mathscr{C}$, a bounded subset of $\mathscr{C}$ that approximates the maximum reachable distance of the character (a $2m$ cube for a character $1.6m$ tall).

Algorithm 2: GROWTREE($\mathscr{T}$, q)

```
q_nearest ← FINDNEAREST(T, q, ρ);
q_root ← q_nearest;
while true do
    if ρ(q_root, q) < ε then q_target ← q;
    else q_target ← q_root + ε(q − q_root);
    if SOLVEIKCONSTRAINTS(q_target) ≠ SUCCESS then
        return FAILED;
    end
    ADDBRANCH(T, q_root, q_target);
    q_root ← q_target;
    if q_target = q then return REACHED;
end
```

After selecting q_{sample}, the function GROWTREE() attempts to grow the tree towards q_{sample} (Algorithm 2). The node $q_{nearest}$ in $\mathscr{T}$ that is nearest to q according to the distance metric ρ is computed. The node $q_{nearest}$ becomes the root of a potential new branch in $\mathscr{T}$. The configuration q_{target} is defined as either the sampled node q itself or as an intermediate node along the straight-line path between the current root node q_{root} and q, depending upon whether the distance between them, $\rho(q_{root},q)$, is greater than a maximum distance threshold ε, referred to as the *RRT step size*. Figure 3 illustrates the process of growing the search tree. Candidate branches are added to $\mathscr{T}$ if and only if the inverse kinematics algorithm in

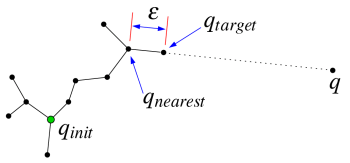


Figure 3: Growing a branch towards a sample configuration q.

SOLVEIKCONSTRAINTS() is able to calculate a valid pose for the character along all the configurations of that branch and no collisions are detected. The planning algorithm terminates either when it exceeds the maximum time allowed t_{max} or when the goal is reached and a complete path between q_{init} and q_{goal} has been found.

3.2 Planning: Computing the Character's Pose

For each object position and orientation tested by the planning algorithm, the corresponding pose of the character is computed using a constrained optimization algorithm for inverse kinematics [Yamane and Nakamura 2003]. The external constraints on the positions of the feet specified by the user and the position of the object specified by the planner (and the resulting position of the hands) are hard constraints. Poses from the motion capture database with similar external constraints are included as soft constraints. The projection of the center of gravity onto the ground is included as a soft constraint so that the character appears balanced. The resulting pose is tested for collisions between the character and the object, between the character and the environment, and between the character's body parts using a public collision detection library ColDet (available at http://photoneffect.com/coldet/).

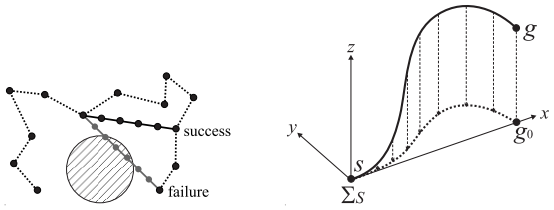
A key component of this process is the posture database which biases the inverse kinematics algorithm toward natural-looking poses via soft constraints. The database contains a set of unique poses, each stored as the 3D positions of 41 markers placed on the subject during a motion capture session (Figure 4). We use marker positions instead of joint angles so that we can use character skeletons with different kinematics and varying degrees of freedom. We captured four box manipulations performed by one subject with different start and goal heights (high to high, high to low, low to high, and low to low). Each motion was captured once and had a duration of six seconds. In a preprocessing step, the database was formed from these captured motions by adding a new pose only if the Euclidean distance between that pose and all the poses currently in the database was greater than a threshold. Before the distance is computed, the marker positions in the two poses are translated and rotated in the horizontal plane so that the squared sum of the marker distances is minimized (effectively removing differences in absolute position and facing direction from the state of the pose).

The planner generates candidate positions and orientations for the object, which are used to find character poses in the database with similar constraints. For example, the constraints might require that the character's hands be 0.5 m in front of the character and vertical to hold the object at the location specified by the planner. Given a database with n_e entries, each with n_m markers and positions of n_c constraints from the planner, our goal is to obtain the weight vector $w \in R^{n_e}$ for the entries in the database. Once w is determined, the position of marker k, p_k, that satisfies the constraints is computed by

$$p_k = \sum_{i=1}^{n_e} w_i p_k^i \qquad (2)$$

where w_i is the i-th element of w and p_k^i denotes the position of marker k in the i-th entry. We assume that each constrained point is

Figure 4: Motion capture subject and marker data.



(a) (b)

Figure 5: (a) Successful and failed smoothing operations; (b) Path representation.

a linear combination of markers, namely,

$$c_m = \sum_{k=1}^{n_m} v_{m,k} p_k \ (m = 1 \ldots n_c) \tag{3}$$

where c_m is the position of the m-th constraint and $v_{m,k}$ is the weight for marker k. Substituting Eq.(2) into Eq.(3) yields

$$\begin{aligned} c_m &= \sum_{k=1}^{n_m} v_{m,k} \left(\sum_{i=1}^{n_e} w_i p_k^i \right) = \sum_{i=1}^{n_e} w_i \left(\sum_{k=1}^{n_m} v_{m,k} p_k^i \right) \\ &= \sum_{i=1}^{n_e} w_i c_m^i = \begin{pmatrix} c_m^1 & c_m^2 & \ldots & c_m^{n_e} \end{pmatrix} w \end{aligned}$$

where c_m^i is the position of constraint m computed from the markers in the i-th entry. We then obtain the following linear equation in w:

$$\begin{pmatrix} c_1^1 & c_1^2 & \ldots & c_1^{n_e} \\ c_2^1 & c_2^2 & \ldots & c_2^{n_e} \\ \vdots & \vdots & & \vdots \\ c_{n_c}^1 & c_{n_c}^2 & \ldots & c_{n_c}^{n_e} \\ 1 & 1 & \ldots & 1 \end{pmatrix} w = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n_c} \\ 1 \end{pmatrix}. \tag{4}$$

The last row is added to constrain the sum of the weights to 1. This equation is solved by using a singularity-robust inverse to compute the weights for the entries [Nakamura and Hanafusa 1986]. The inverse can be precomputed to reduce the time required to compute the desired marker positions during planning.

3.3 Postprocessing: Smoothing

The planner is able to operate in a high-dimensional space because of the heuristic search strategy. However, the generated path is not necessarily short or smooth. The system shortens and smooths the path by iteratively selecting two nodes along the solution path and attempting to connect them by interpolating in configuration space (Figure 5(a)). Although more sophisticated techniques for smoothing are possible, this simple scheme suffices. Given two selected path points, we linearly interpolate the translation components and use quaternion spherical linear interpolation (slerp) for the rotation [Shoemake 1985]. The inverse kinematics algorithm is used to calculate the full body pose for each configuration along the new, shorter path and the system checks for collisions among the character, environment, and object. If a collision is detected, the planner rejects the shorter path and the original path is restored. If no collision is found, the shorter path replaces the original path. This process is repeated until no further progress in shortening the path can be made.

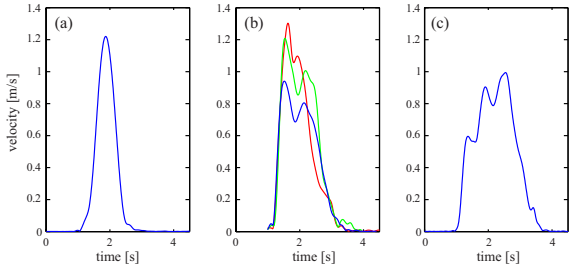


Figure 6: Recorded velocity profiles from human subjects: (a) a simple pick and place task that has the expected bell-shaped velocity curve; (b) three pick and place tasks where the human subject also had to avoid a pole in the middle of the workspace (which causes a dip in the velocity profile); (c) a compound task where the human subject had to avoid a table as well as a pole.

3.4 Postprocessing: Velocity Profile

The path computed by the planner is turned into a trajectory by using a model velocity profile of human motion. We use a velocity profile model derived from a database of 25 pick-and-place manipulation trajectories and apply it to the path. Velocity profiles for human arm motion have very characteristic shapes. Numerous detailed studies of arm movements in the psychophysical literature found the motion to be extremely stereotypical. For example, for point to point arm movements, Atkeson and Hollerbach [1985] tried several different methods to disrupt subjects and they still used bell-shaped velocity profiles for this simple task. Figure 6 shows a similar pattern for the velocity profiles in our database. The system selects a model path closest to the smoothed path from the trajectory database and then uses its velocity profile as a template for the synthesized trajectory.

To represent the path independently of the velocity profile, we use two functions for the path in the horizontal plane and a third for the path in the vertical direction. The intuition behind this decomposition is that the dynamics of the motion in the vertical and horizontal directions are different due to gravity. Each of the four functions (three for the path and one for the velocity profile) is approximated by a set of radial basis functions (RBF). This representation is also used to compare the current path with paths in the database to determine the most appropriate velocity profile.

Let s and g denote the start and goal positions respectively and g_0 denote the projection of g onto the horizontal plane that includes s, and consider a frame Σ_S whose origin is at s, with the z axis vertical, and the x axis pointing toward g_0 (Figure 5(b)). First, we obtain $(N+1)$ points $p_0 \ldots p_N (p_0 = s, p_N = g)$ on the path by interpolating the points of the nodes generated by the planner, all represented in Σ_S. Let the *traveled distance* u_k of point k be defined as:

$$u_k = \sum_{i=1}^{k} |p_i - p_{i-1}| \tag{5}$$

where $u_0 = 0$ and u_N equals the total traveled distance l. The *normalized traveled distance* of point k is computed by $\bar{u}_k = u_k / l$. We

then compute the difference between the maximum and minimum values in each direction $r_* = p_*^{max} - p_*^{min}$ $(* = x,y,z)$. The normalized value $\bar{p}_{k*}$ in each direction is defined as:

$$\bar{p}_{k*} = \frac{1 - e^{-cr_*}}{r_*} p_{k*} \ (* = x,y,z) \tag{6}$$

where c is a positive constant. We can approximate the functions $\bar{p}_*(\bar{u})$ $(* = x,y,z)$ by a combination of weighted RBFs:

$$\bar{p}_*(\bar{u}) = \sum_{i=1}^{m} w_{*i} \Phi_i(\bar{u}) \tag{7}$$

$$\Phi_i(\bar{u}) = \exp\left(-\frac{(\bar{u} - a_i)^2}{2\sigma^2}\right) \tag{8}$$

where m is the number of RBFs and $a_i (1 \leq i \leq m)$ is the center of the i-th RBF, and σ is the gaussian standard deviation. The normalized velocity $\bar{v}$ is represented as a function of the normalized traveled distance $\bar{u}$. This function is also approximated by a set of RBFs:

$$\bar{v}(\bar{u}) = \sum_{i=1}^{m} w_{ui} \Phi_i(\bar{u}) \tag{9}$$

The smoothed path is interpolated, normalized, and approximated in the same way and the weights for the RBFs are computed. The weight vector is compared with the entries in the trajectory database and the closest trajectory is selected. Let T_{DB} and L_{DB} denote the duration and length of the closest trajectory respectively. We also know the length of the smoothed path L_P. The velocity profile of the closest trajectory is applied to the smoothed path by integrating $\bar{v}$ to compute the velocity profile of the trajectory:

$$\bar{u}(\bar{t}_i) = \bar{u}(\bar{t}_{i-1}) + \bar{v}(\bar{u}(\bar{t}_{i-1}))\Delta\bar{t} \ , \ (i \geq 1) \tag{10}$$

where $\Delta\bar{t}$ is a small time step, $\bar{t}_i = i\Delta\bar{t}$, and $\bar{u}(0) = 0$. Eq.(10) is repeated until $\bar{u}(t_i)$ is greater than 1. Let $\bar{T}_{DB}$ denote the time at the last step. The duration of the planned motion T_P is computed by:

$$T_P = T_{DB}\bar{T}_{DB}\frac{L_P}{L_{DB}} \tag{11}$$

This equation is based on the hypothesis that the average velocity of the planned motion should be roughly the same as the closest match in the trajectory database. Finally, the velocity profile is computed:

$$u(t_i) = L_P \bar{u}(\bar{t}_i), \ t_i = T_{DB}\frac{L_P}{L_{DB}}\bar{t}_i \tag{12}$$

The position at time t is obtained by finding the point on the smoothed path where the traveled distance matches $u(t)$.

3.5 Gaze Synthesis

Eye motion is very important for human animation because the eyes are a key component of nonverbal communication [Lee et al. 2002b]. Unfortunately, most motion capture systems do not include gaze tracking and it is not part of the motion capture database used in this paper. Instead, we rely on an approximate model derived from biomechanical observations. Given a visual target, the eyes and head move simultaneously to form a stable, directed gaze. The eye movements rotate the optic axis with respect to the head so that the visual target is either acquired or maintained in the central area of the retina [Morasso and Tagliasco 1986]. The movement of the eye-neck chain facilitates the visual feedback necessary for accuracy in executing a given task (hand-eye coordination). This coordination implies a geometric mapping between the "global" task point trajectory and the "internal" joint variables of the eye-neck chain. In general, such a mapping is nonlinear, but observations of people have revealed an approximately linear mapping [Morasso and Tagliasco 1986].

Based on these models, we have devised an approximate *gaze function* to compute eye movements for manipulation tasks. A point of interest (POI) defines the desired gaze direction. A different POI is used for each segment of the computed motion. While the character is reaching for an object, the POI is the location of the object. After the object has been acquired, the POI becomes the destination location for the object. The motion of the head and eyes overlap with the motion of the arms. Based on our experiments, the gaze motion should be initiated at the same time or slightly before the arm motion. All of the examples in the paper use this gaze model. A flaw with this approach is that the gaze direction is sometimes too precise; a human performing the same task would not need to focus so closely on the object and goal. A biomechanically based noise model might help to further increase the naturalness.

4 Results

This section presents animations synthesized for a variety of manipulation tasks. Figure 7 presents a collection of object start and goal locations. Our planner can automatically synthesize manipulation motions for any combination of these start and end positions. Figure 8 illustrates a series of animations synthesized for two-hand pick-and-place manipulation tasks. All animations used the same pose database (257 total postures). The top row, (a), shows a synthesized motion for a clown repositioning a box. The next six rows demonstrate variation in character kinematics, object shapes, environments, and tasks. The images in sequences (b) and (c) show a clown character with very long legs and a gorilla character with long arms moving a box. Sequences (d) and (e) illustrate synthesized animations for a long, thin object in environments where the object path is not obvious. The planner can still automatically generate a collision-free motion. Sequences (f) and (g) contain examples of planned output for entirely different tasks (opening a desk drawer and a cabinet). Sequence (h) shows a motion of an adult and a child cooperating to carry a large object. All of these examples used the same posture database, which was captured from a human subject whose kinematics differed from each of these characters. These examples illustrates the ability of the algorithm to handle large variations in the character kinematics because scaled marker locations are represented as soft constraints. Finally, we demonstrate a complicated compound task: opening a desk drawer, removing an object from a shelf, inserting it into the drawer, and closing the drawer (Figure 9). This entire motion was automatically synthesized by sequencing the output from seven invocations of the planner, one for each different set of contacts.

Most of the computation time was spent performing the inverse kinematics and constrained optimization calculations. Table 1 summarizes the computation time required for creating one node with four different sets of constraints, measured on a 2.0GHz Pentium IV PC. The inverse kinematics computation is iterated an average of 7–10 times at each node to meet the given error tolerance. The planning for this setting typically generates 200-300 nodes in the search tree. This results in totals of approximately 21 to 32 seconds with both data and balance constraints. Although, the running time of the planner primarily depends on the difficulty of simultaneously satisfying all of the required constraints, it also depends on the size of the database. For the database we are currently using (257 poses), the time to calculate the nearest matching pose is relatively small even with a brute-force $O(n)$ algorithm. A larger database may require an $O(\log n)$ algorithm that uses efficient nearest-neighbor techniques from computational geometry, for example, multi-dimensional k-d trees [Arya et al. 1998].

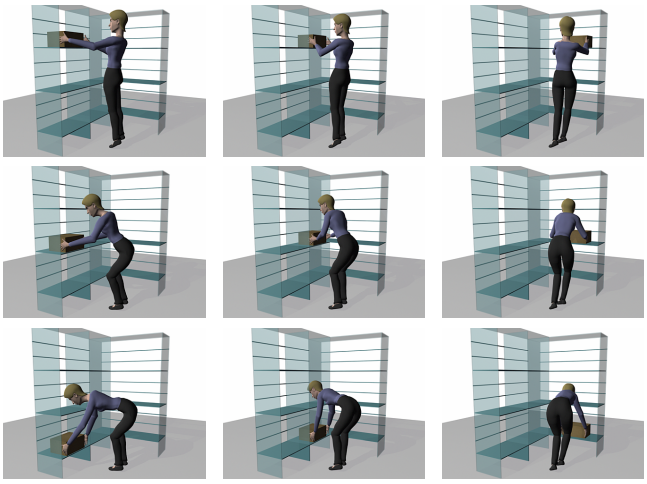


Figure 7: A collection of nine initial and goal postures, any of which can be used in the synthesis of new motion.

data	yes	no	no	yes
balance	yes	no	yes	no
total	107	64	68	103
marker	14	—	—	13
IK	93	64	68	90

Table 1: Computation time per node for computing marker positions and inverse kinematics [*ms*].

5 Discussion

We have developed an algorithm for animating whole-body motions for human-like characters that relies on constrained inverse kinematics, path planning, and a library of captured example postures. The algorithm synthesizes natural-looking motions for manipulation tasks while respecting both environment and posture constraints. The work required to generate a sequence of motion, in addition to running the planning program, is (1) set the start and goal position/orientation of the object, and (2) set the initial configuration of the character. The time for this procedure would be in the order of a few minutes with the interface of typical 3D CG software packages. The total of man hours and CPU time is therefore under 10 minutes, while even an expert artist would likely take an hour to keyframe similar motions.

One advantage of our approach is that the data need not be a perfect match for the task. Because the system is model-based, the knowledge about human motion inherent in those models allows the search process of the planner and inverse kinematics algorithm to fill in some gaps. For example, Figure 8 presents examples in which the object being manipulated is substantially different than the one used in the motion capture sessions. The shape of the object is taken into account during collision detection and via the user's specification of the grasp points. The weight of the object affects the posture of the character so that it continues to appear balanced. If however, human strategies for lifting heavy objects are fundamentally different than those for lifting light ones ("bend with your knees not with your back," for example), those strategies will appear only if the database includes sequences in which the actor used that strategy. A more detailed biomechanical model could also be used to create this particular effect by taking into account the forces in the back during a lifting task.

A major limitation of the method is that the system will fail if the goal is unreachable or the character cannot remain balanced. Extending an arm for balance is also a solution that will not be found unless such a pose is included in the database. Basically the "strategy" used for the task should be represented in the database in some way for the system to succeed. Additional data with different characteristics can also increase the variety of motions if they are stored in a different database and the planner has the ability to search among multiple databases. For example, additional databases could be used to synthesize collision-free motions in a complex environment by providing collision-avoidance behaviors (e.g., lower the head) if character-environment collisions cause failure in the planning phase using a single database.

We do not yet have a way of measuring when the data is appropriate to the task or how much data is sufficient to produce natural-looking motion for a given task. For example, we successfully use the recorded motion of placing boxes on shelves to compute the motion of a character performing quite dissimilar tasks such as opening a desk drawer and a cabinet. We have also observed situations in which a very small and dissimilar data set degraded the quality of the motion by providing inappropriate examples. In our experiments, additional data always improved the quality of the motion provided that the motion in the database contained postures in which the constraints on the limbs were a subset of the constraints on the limbs for the motion being created. For example, synthesizing a motion for a one-handed manipulation task using data collected for a two-arm manipulation task might cause unnecessary and unnatural motion of the free arm.

When computing the velocity profiles for synthesized motions, we utilize the profiles for similar motions contained in the database. Although all of the profiles in the database had characteristic bell-shaped curves, subtle dips in the velocity due to obstacles in the workspace were not directly associated with our synthesized examples. Using minimum distance information to relate obstacle clearance with object velocity could potentially provide an automatic way to synthesize obstacle-induced velocity dips.

The manipulation tasks that we explored were all large scale and the grasps of the objects were simple. Our motion capture database included only three markers on the hands. To include more complex grasps in the planning phase, we would likely need data for the detailed motion of the fingers during the grasp. One potential advantage of our approach is that the data used to bias the planning process can come from multiple sources. We may, therefore, be able to avoid the problem faced by motion capture teams who must capture motion at vastly different scales in one shoot (hands and whole body, for example) by capturing similar motions at different resolutions in two separate shoots and integrating the two sets of data via the planning algorithm. Similarly, we might be able to include eye gaze information recorded with an eye tracker rather than the model-based approach used here.

Although we have explored only manipulation tasks to date, this algorithm should be easily extended to other highly constrained motions such as sitting down in a chair and getting into or out of a vehicle or cockpit. Extensions to operations that involve regrasping or intermittent contact will prove more difficult because of the need for additional models of how humans select from among a collection of possible grasps.

Acknowledgments

This research was supported in part by NSF Award ANI-0203912, NSF Award IIS-0196089 and the Digital Human Research Center (AIST, Japan). The authors gratefully acknowledge Moshe Mahler and Mike Stevens for help with Maya modeling and rendering. We also thank Chris Atkeson, Steven Osman, the graphics group at Georgia Tech and the anonymous reviewers for their helpful comments.

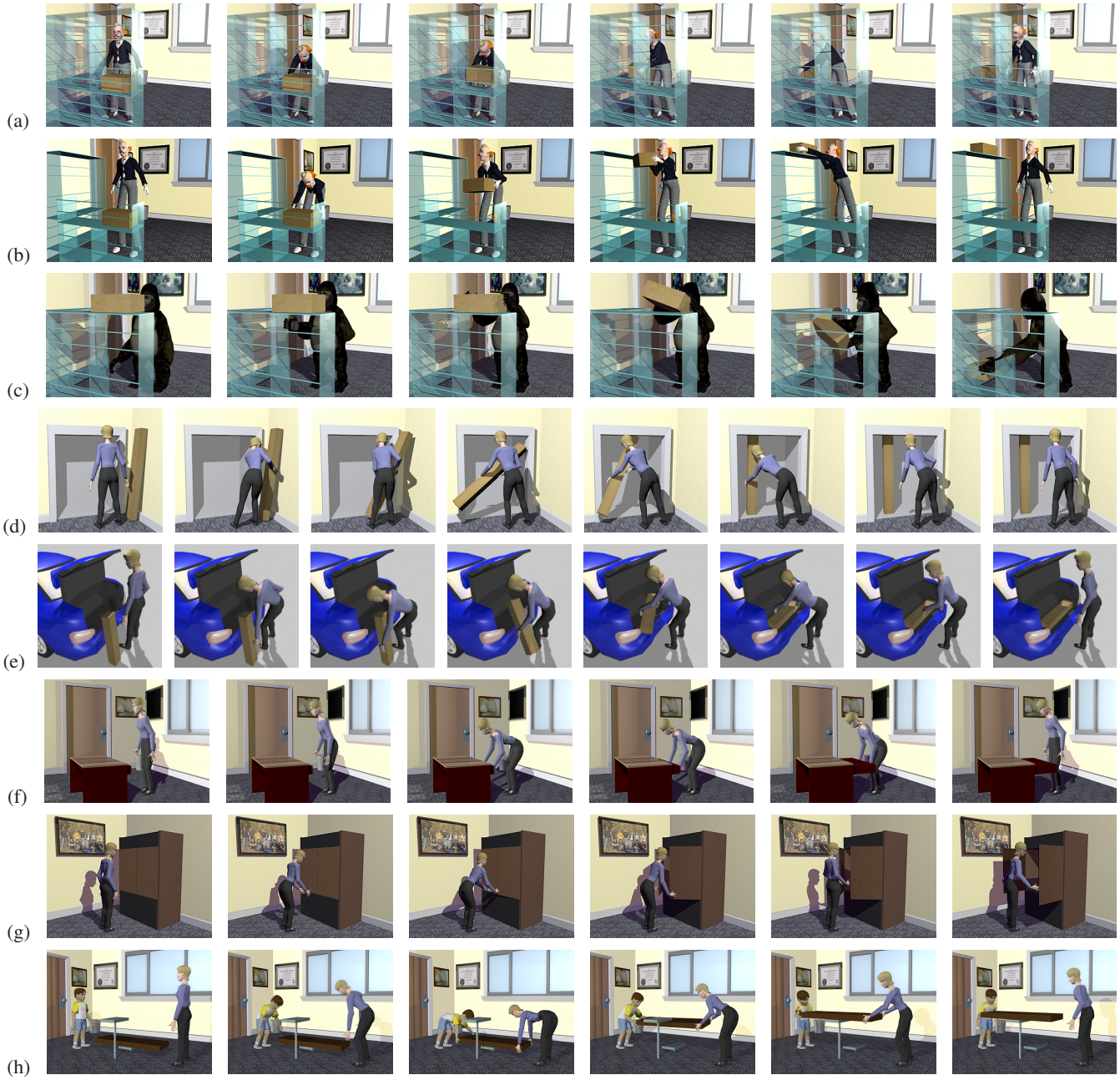


Figure 8: Synthesized motion for a variety of characters, objects, and tasks: (a) clown moving a box with two hands; (b) clown with very long legs; (c) gorilla with long arms; (d) manipulating a long thin vertical object; (e) inserting a box into a car trunk; (f) opening a desk drawer; (g) opening a cabinet; (h) cooperative manipulation of a single object by two characters with different geometry and kinematics.

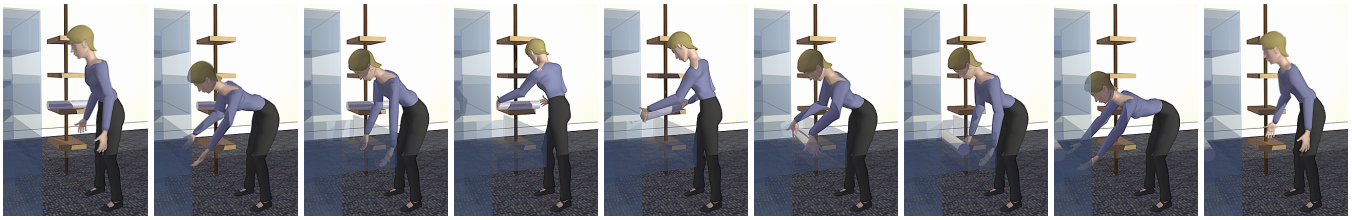


Figure 9: Compound example: automatically synthesized motion to open a desk drawer, retrieve an object from a shelf, insert it into the drawer, and close the drawer (a total of seven planned paths).

References

ARIKAN, O., AND FORSYTH, D. A. 2002. Synthesizing constrained motions from examples. *ACM Trans. on Graphics 21*, 3, 483–490.

ARIKAN, O., FORSYTH, D. A., AND O'BRIEN, J. F. 2003. Motion synthesis from annotations. *ACM Trans. on Graphics 22*, 3, 402–408.

ARYA, S., MOUNT, D. M., NETANYAHU, N. S., SILVERMAN, R., AND WU, A. Y. 1998. An optimal algorithm for approximate nearest neighbor searching. *Journal of the ACM 45*, 891–923.

ATKESON, C., AND HOLLERBACH, J. 1985. Kinematic features of unrestrained vertical arm movements. *Journal of Neuroscience*, 9, 318–2330.

BANDI, S. 1998. *Discrete Object Space Methods for Computer Animation*. PhD thesis, Swiss Federal Institute of Technology.

BARRAQUAND, J., AND LATOMBE, J.-C. 1990. Robot motion planning: A distributed representation approach. *Int. Journal of Robotics Research 10*, 6 (December), 628–649.

BINDIGANAVALE, R., GRANIERI, J., WEI, S., ZHAO, X., AND BADLER, N. 1994. Posture interpolation with collision avoidance. In *Proc. of Computer Animation '94*, 13–20.

BOULIC, R., MAS, R., AND THALMANN, D. 1997. Complex character positioning based on a compatible flow model of multiple supports. *IEEE Trans. on Visualization and Computer Graphics* (July-Sept), 245–261.

BRUDERLIN, A., AND CALVERT, T. W. 1989. Goal-directed, dynamic animation of human walking. In *Computer Graphics (Proc. SIGGRAPH 89)*, vol. 23, 233–242.

CHING, W., AND BADLER, N. 1992. Fast motion planning for anthropometric figures with many degrees of freedom. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, 2340–2345.

GLEICHER, M. 1997. Motion editing with spacetime constraints. *1997 Symposium on Interactive 3D Graphics*, 139–148.

GLEICHER, M. 1998. Retargeting motion to new characters. In *Proc. SIGGRAPH 98*, 33–42.

JUNG, M. R., BADLER, N., AND NOMA, T. 1994. Animated human agents with motion planning capability for 3D-space postural goals. *J. Visualization and Comp. Animation 5*, 4, 225–246.

KALISIAK, M., AND VAN DE PANNE, M. 2000. A grasp-based motion planning algorithm for character animation. In *Proc. Comp. Animation and Simulation 2000*, Springer-Verlag, 43–58.

KAVRAKI, L., ŠVESTKA, P., LATOMBE, J. C., AND OVERMARS, M. H. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration space. *IEEE Trans. on Robotics and Automation 12*, 4, 566–580.

KAWATO, M. 1999. Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology 9*, 718–727.

KOGA, Y., KONDO, K., KUFFNER, J., AND LATOMBE, J.-C. 1994. Planning motions with intentions. In *Proc. SIGGRAPH 94*, 395–408.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. *ACM Trans. on Graphics 21*, 3, 473–482.

KUFFNER, J., KAGAMI, S., NISHIWAKI, K., INABA, M., AND INOUE, H. 2002. Dynamically-stable motion planning for humanoid robots. *Autonomous Robots 12*, 105–118.

KUFFNER, J. 1999. *Autonomous Agents for Real-time Animation*. PhD thesis, Stanford University.

LATOMBE, J. C. 1991. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA.

LAVALLE, S., AND KUFFNER, J. 2001. Randomized kinodynamic planning. *Int'l Journal of Robotics Research 20*, 5, 378–400.

LEE, J., CHAI, J., REITSMA, P. S. A., HODGINS, J. K., AND POLLARD, N. S. 2002. Interactive control of avatars animated with human motion data. *ACM Trans. on Graphics 21*, 3, 491–500.

LEE, S., BADLER, J., AND BADLER, N. 2002. Eyes alive. *ACM Trans. on Graphics 21*, 3, 637–644.

LIU, Y., AND BADLER, N. I. 2003. Real-time reach planning for animated characters using hardware acceleration. In *Proc. Computer Animation and Social Agents (CASA)*, 86–93.

MORASSO, P., AND TAGLIASCO, V., Eds. 1986. *Human Movement Understanding: from computational geometry to artificial intelligence*. North-Holland.

NAKAMURA, Y., AND HANAFUSA, H. 1986. Inverse Kinematics Solutions with Singularity Robustness for Robot Manipulator Control. *J. Dynamic Sys., Meas., and Control 108*, 163–171.

PARK, W., CHAFFIN, D., AND MARTIN, B. 2002. Modifying motions for avoiding obstacles. *SAE Transactions 110*, 6, 2250–2256.

PETTRE, J., LAUMOND, J.-P., AND SIMEON, T. 2003. A 2-stages locomotion planner for digital actors. In *ACM SIGGRAPH Symp. on Computer Animation*, 258 – 264.

ROSE, C. F., GUENTER, B., BODENHEIMER, B., AND COHEN, M. F. 1996. Efficient generation of motion transitions using spacetime constraints. In *Proc. SIGGRAPH 96*, 147–154.

ROSE, C., COHEN, M. F., AND BODENHEIMER, B. 1998. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics & Applications 18*, 5, 32–40.

ROSE, C. F., SLOAN, P.-P. J., AND COHEN, M. F. 2001. Artist-directed inverse-kinematics using radial basis function interpolation. *Computer Graphics Forum 20*, 3, 239–250.

SHOEMAKE, K. 1985. Animating rotation with quaternion curves. In *Proc. SIGGRAPH 85*, 245–254.

SIMEON, T., CORTES, J., SAHBANI, A., AND LAUMOND, J. P. 2002. A manipulation planner for pick and place operations under continous grasps and placements. In *Proc. Int'l Conf. on Robotics and Automation*, 2022–2027.

WILEY, D., AND HAHN, J. 1997. Interpolation synthesis of articulated figure motion. *IEEE Computer Graphic and Applications 17*, 6, 39–45.

YAMANE, K., AND NAKAMURA, Y. 2003. Natural Motion Animation through Constraining and Deconstraining at Will. *IEEE Trans. on Visualization and Computer Graphics 9*, 3, 352–360.