8.7 Interactive Motion Correction and Object Manipulation

Interactive Motion Correction and Object Manipulation

Ari Shapiro[*]
University of California, Los Angeles

Marcelo Kallmann[†]
Computer Graphics Lab
University of California, Merced

Petros Faloutsos[‡]
University of California, Los Angeles

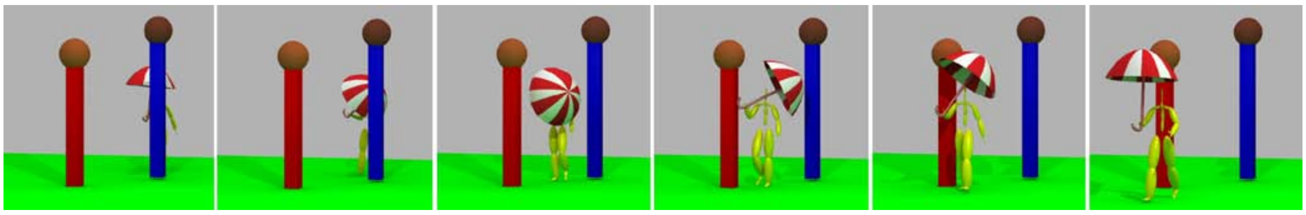

Figure 1: In order to avoid collisions between the umbrella and the two posts the arm motion was planned in sync with a walking sequence.

Abstract

Editing recorded motions to make them suitable for different sets of environmental constraints is a general and difficult open problem. In this paper we solve a significant part of this problem by modifying full-body motions with an interactive randomized motion planner. Our method is able to synthesize collision-free motions for specified linkages of multiple animated characters in synchrony with the characters' full-body motions. The proposed method runs at interactive speed for dynamic environments of realistic complexity. We demonstrate the effectiveness of our interactive motion editing approach with two important applications: (a) motion correction (to remove collisions) and (b) synthesis of realistic object manipulation sequences on top of locomotion.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques.

Keywords: character animation, motion capture, motion editing, virtual humans, object manipulation

1 Introduction

Techniques based on libraries of motion capture produce the most realistic animations of virtual humans to date. However one of the main drawbacks of such techniques is their inability to offer, without additional mechanisms, any variations from the exact recorded motions. The virtual environments where the playback of motion occurs differ from the environment in which the motion was captured. Virtual environments often contain obstacles and characters that were not present in the motion capture studio. The motion capture data must be modified to accommodate the virtual environment in order to preserve the appearance of realism. For example, a virtual human may need to swing its arm away to avoid a virtual object or lift its leg higher in order to step over an obstacle that lies on the ground.

[*]e-mail: ashapiro@cs.ucla.edu
[†]e-mail:mkallmann@ucmerced.edu
[‡]e-mail:pfal@cs.ucla.edu

Designing in advance all required motions for a given virtual environment or scenario involves tedious and time-consuming design work. Furthermore, it is not practical to rely on pre-designed motions when object grasping and manipulation are required for arbitrarily located objects in arbitrary scenes. A recorded motion captured of a person catching a ball with two hands at chest-level will not be effective for catching a different sized ball with one hand at waist-level. The problem is even more complex when the character, the target and the obstacles in the environment move. This is the problem that we address in this work.

We introduce a new motion editing approach that combines recorded motions with motion planning in order to produce realistic animations of characters avoiding and manipulating moving objects in dynamic environments. The approach has two applications: 1) *Motion correction*, where a prerecorded motion played on a virtual human is automatically corrected to respect obstacles in the virtual environment, and 2) *Object manipulation*, where virtual humans are instructed to grab, drop and touch various objects, either moving or fixed, while playing back recorded motion and respecting both fixed and moving obstacles As an example of motion correction, Figure 1 shows a walking character manipulating an umbrella so that it can walk through the two posts without hitting them. The motion of the character's arms was synthesized interactively by our planner on top of the original locomotion.

Our approach is based on a motion planner that generates collision-free motions for specified linkages of a character, making them reach desired targets at specified times. In order to address the time constraints, the planner considers time as one additional dimension in the search space. Therefore moving targets, moving obstacles and synchronization with keyframe animations can all be taken into account together. Our method proves to be very efficient for producing object manipulation sequences as well as for adjusting motions to avoid collisions with obstacles. We are also able to control stylistic aspects of the resulting motions by customizing the search heuristics that our planner employs when exploring the space of possible configurations. Furthermore, we employ an anatomically meaningful skeleton parameterization that helps us enforce realistic limits on the motion of the character's joints. To demonstrate the effectiveness of our approach, we present several complex examples solving highly dynamic tasks.

Contributions. We present: (a) a hybrid motion synthesis solution that combines recorded motions with motion planning, and (b) how to control the obtained results by choosing different configuration sampling strategies for the motion planner.

In general our method can be applied to generic task-oriented motion editing problems respecting collision-free and spatio-temporal constraints. It requires about one second of time to compute motions of average complexity making it suitable for interactive use.

2 Related Work

Motion planning research for animated characters has traditionally been segmented into two different areas; 1) full-body motion planning for the purpose of locomotion, and 2) reach and arm planning for the purpose of object manipulation.

Full-body locomotion planning. Motion synthesis, whose primary goal is to generate plausible motion that adheres to given constraints, such as a movement path, has been explored by past research [Lau and Kuffner 2005; Kwon and Shin 2005; Lai et al. 2005; Kovar et al. 2002; Arikan and Forsyth 2002; Choi et al. 2002]. The goal of our method differs in that rather than generating locomotion sequences as these other methods do, it targets arm and leg movements corrections that either adhere to constraints in the virtual environments or allow object manipulations not included in the original motion. Our method uses the time dimension in planning in order to handle dynamic obstacles. Of particular note, [Lau and Kuffner 2005] uses the time dimension in order to plan for dynamic obstacles. In contrast, our method uses the time dimension in concert with a locomotion clip to handle object manipulation and arm linkage adjustments, rather than to generate the underlying locomotion clip. Also, our method does not require a preprocessed set of motion clips that have already been segmented and transformed into an FSM, and can use any motion clip that propels the animated character. The inclusion of the time dimension in a motion planner has also already been proposed in Robotics [Hsu et al. 2002]; however, we present a planner that uses the time dimension for a specific set of tasks; to plan the motions of some limbs in synchronization with external motions affecting the same character.

Given the complementary nature of our work to the research in locomotion synthesis, our method could be enhanced with the inclusion of such methods as a preliminary motion editing stage. The locomotion generation method would create a motion clip, which would subsequently be used as input into our system which would in turn edit and arm and leg movements of the resulting animation.

Of relation to our work, [Pettré et al. 2003] uses a two-stage locomotion planner to first plan the movement of the character, then correct the upper body for collisions. Our method differs in it can handle simultaneously moving targets, moving obstacles and moving characters.

Reach and arm planning Since the first application of motion planning to computer animation [Koga et al. 1994] which included grasp planning, several motion planning methods have been proposed specifically addressing human-like characters manipulating objects.

One approach is to search for a sequence of intermediate suitable hand locations in the workspace and use Inverse Kinematics (IK) to derive arm postures reaching each intermediate hand location [Yamane et al. 2004; Liu and Badler 2003; Bandi and Thalmann 2000]. The final valid motion is obtained through interpolation of the postures. Another approach is to search directly in the configuration space [Koga et al. 1994; Kuffner and Latombe 2000], yielding simpler algorithms (not requiring IK during the search) that can address the entire solution space. As the search space grows exponentially with its dimension, simplifying control layers can be specified for synthesizing whole-body motions [Kallmann et al. 2003].

Hardware acceleration has been used to generate arm linkage paths for manipulation purposes [Liu and Badler 2003] for stationary characters. Our work is similar in that we use a similar analytically-based IK algorithm [Tolani and Badler 1996], however our method works with both non-stationary characters as well as moving objects.

A key feature of our method is efficiency. We choose to perform the search in the configuration space, relying on a *Rapidly-Exploring Random Tree* (RRT) planner [LaValle 1998; LaValle and Kuffner 2000] in its bidirectional version [Kuffner and LaValle 2000] along with adding the time dimension to the search. This allows our method to be used interactively by an animator.

The problem of synthesizing human-like arm movements is addressed by [Yamane et al. 2004] by using examples from motion capture examples to generate velocity profiles of natural arm motions. Unlike this method, our method is able to plan motions that involve moving feet and moving characters. Our method does not use example motions and thus our arm movements are less likely to look as natural. However, we are able to generate a solution with much greater speed, on the order of seconds rather than minutes, and thus are much better suited for interactive use.

3 Problem Formulation

We represent the character as an articulated figure composed of hierarchical joints. Let C^F be the space of all full configurations of the character. Let $c^f = (p, q^1, \ldots, q^{r-1}, q^r, \ldots, q^n) \in C^F$ be one full configuration of the character, where $p \in \mathbb{R}^3$ is the position of the root joint, and $q^i \in \mathbb{S}^3, i \in \{1, \ldots, n\}$, is the rotational value of the i^{th} joint, in quaternion representation. The components of c^f are organized in such way that the rear part $c^p = (q^r, \ldots, q^n) \in C^P$ denotes the degrees of freedom (DOFs) controlled by the planner, and the fore part $c^m = (p, q^1, \ldots, q^{r-1}) \in C^M$ contains the remaining DOFs, which are controlled by an external motion. Therefore $C^F = C^M \times C^P$ and $c^f = (c^m, c^p)$.

An external motion controller affecting the DOFs in C^M is defined as a time-varying function $m^m(t) = (p(t), q^1(t), \ldots, q^{r-1}(t))$. Therefore $p(t) \in \mathbb{R}^3$ describes the translational motion of the root joint, and $q^i(t) \in \mathbb{S}^3$, $1 \leq i < r$, describes the rotational motion of the affected joints in their local frame coordinates. We assume that m^m is completely defined over a given time interval $I \subset \mathbb{R}$, as is the case for motions defined as keyframe animations. We furthermore assume that $m^m(t)$ is collision-free for all $t \in I$.

In order to take into account moving objects in the environment, all object motions are required to be parameterized by the same time parameter $t \in I$ of motion m^m. We therefore construct a function $w(t)$, which sets the state of the world to the desired time t.

Let $c^p_{init} \in C^P$ and $c^p_{goal} \in C^P$ be initial and goal configurations specified to be reached at times t_{init} and t_{goal} respectively, $[t_{init}, t_{goal}] \subset I$. Our search space includes the time dimension and is defined as $C^S = C^P \times [t_{init}, t_{goal}]$. Configuration $c^s = (c^p, t) \in C^S$ is *valid* if the character's posture $(m^m(t), c^p) \in C^F$ respects joint limits and is collision-free when the world's state is $w(t)$. We denote by C^S_{free} the subspace of all valid configurations in C^S.

Consider now $c^s_{init} = (c^p_{init}, t_{init})$ and $c^s_{goal} = (c^p_{goal}, t_{goal})$ be initial and goal configurations in C^S_{free}. Our problem is then reduced to finding a path in C^S_{free} connecting c^s_{init} to c^s_{goal}.

Our planner solves the problem by searching for a sequence of valid landmarks $c^s_i = (q^r_i, \ldots, q^n_i, t_i) \in C^S$, $1 \leq i \leq k$, such that:

1. $c^s_1 = c^s_{init}$, and $c^s_k = c^s_{goal}$,
2. the time parameter is monotone, i.e., $t_i < t_{i+1}$, $1 \leq i < k$,
3. for all pairs of adjacent landmarks (c^s_i, c^s_{i+1}), $1 \leq i < k$, the motion obtained through interpolation between c^s_i and c^s_{i+1} remains in C^S_{free}.

Let q^j_i, $r \leq j \leq n$, be the j^{th} quaternion of landmark c^s_i, $1 \leq i \leq k$. Motion $m^p(t)$ can then be constructed as:

$$m^p(t) = (q^r(t), \ldots, q^n(t)),$$
$$\text{with } q^j(t) = slerp(q^j_i, q^j_{i+1}, \tfrac{(t-t_i)}{t_{i+1}-t_i}),$$
$$\text{and } t_i \leq t < t_{i+1}.$$

The composite motion $m^f(t) = (m^m(t), m^p(t))$, $t \in [t_{init}, t_{goal}]$, will be a valid motion satisfying constraints c^p_{init} and c^p_{goal} at times t_{init} and t_{goal} respectively, and therefore solving our problem. We present in the following section our motion planner, which finds the sequence of landmarks c^s_i required for constructing $m^p(t)$.

4 Synchronized Motion Planner

The goal of our planner is to find a sequence of landmarks connecting c^s_{init} to c^s_{goal} in C^S_{free}. For solving this problem we propose a bidirectional RRT planner algorithm that supports landmarks with monotone time parameters.

4.1 Algorithm

Algorithm 1 summarizes our implementation. Two search trees T_{init} and T_{goal} are initialized having c^s_{init} and c^s_{goal} respectively as root nodes, and are sent to the planner. The trees are iteratively expanded by adding valid landmarks. When a valid connection between the two trees can be concluded, a successful path in C^S is found. Otherwise when a given amount of time has passed, the algorithm fails.

Algorithm 1 SYNCPLANNER (T_1, T_2)

```
 1: while elapsed time ≤ maximum allowed time do
 2:     $c^s_{sample}$ ← SAMPLECONFIGURATION().
 3:     $c^s_1$ ← closest node to $c^s_{sample}$ in $T_1$.
 4:     $c^s_2$ ← closest node to $c^s_{sample}$ in $T_2$.
 5:     if INTERPOLATIONVALID $(c^s_1, c^s_2)$ then
 6:         return MAKEPATH $(root(T_1), c^s_1, c^s_2, root(T_2))$.
 7:     end if
 8:     $c^s_{exp}$ ← NODEEXPANSION $(c^s_1, c^s_{sample}, \varepsilon)$.
 9:     if $c^s_{exp} \neq null$ and INTERPOLATIONVALID $(c^s_{exp}, c^s_2)$ then
10:         return MAKEPATH $(root(T_1), c^s_{exp}, c^s_2, root(T_2))$.
11:     end if
12:     Swap $T_1$ and $T_2$.
13: end while
14: return failure.
```

Line 2 in algorithm 1 requires a sampling routine in C^S for guiding the search for successful landmarks. Our sampling routine is customized for human-like characters and is explained in detail in section 4.2.

Lines 3 and 4 require searching for the closest configurations in each tree. A linear search suffices as the trees are not expected to grow much. The metric used is a weighted sum of time and arm posture metrics. Let c^s_1 and c^s_2 be two configurations in C^S, such that $c^s_j = (q^r_j, \ldots, q^n_j, t_j)$, $j \in \{1, 2\}$. Let p^i_j be the position (in global coordinates) of the joint affected by rotation q^i_j, $r \leq i \leq n$. The distance between c^s_1 and c^s_2 is computed as:

$$dist(c^s_1, c^s_2) = w_t |t_1 - t_2| + w_a \max_i \|p^i_1 - p^i_2\|,$$

where w_t and w_a are the desired weights.

Lines 5 and 9 check if the interpolation between two configurations is valid. It is considered valid if two tests are successful:

1. the configuration in T_{init} has to have its time component smaller than the configuration in T_{goal},
2. the interpolation has to remain in C^S_{free}.

The simplest approach for testing item 2 above is to perform several discrete collision checks along the interpolation between the two configurations. In order to promote early detection of collisions, we use the popular recursive bisection for determining where to perform the discrete tests, until achieving a desired resolution. Note that continuous tests not requiring a resolution limit are available and can be integrated [Schwarzer et al. 2002].

The algorithm tests at lines 5 and 9 if a valid connection between T_1 and T_2 has been found, and in such cases a path in C^S is computed and returned as a valid solution. The path is computed using routine MAKEPATH$(c^s_1, c^s_2, c^s_3, c^s_4)$ (lines 6 and 10), which connects the tree branch joining c^s_1 with c^s_2 to the tree branch joining c^s_3 with c^s_4, with the path segment obtained with the interpolation between c^s_2 and c^s_3.

The node expansion in line 8 uses c^s_{sample} as growing direction and computes a new configuration c^s_{exp} as follows:

$$c^s_{exp} = interp(c^s_1, c^s_{sample}, t), \text{ where}$$
$$t = \varepsilon / d,\ d = dist(c^s_1, c^s_{sample}).$$

Null is returned in case the expansion is not valid, i.e. if the interpolation between c^s_1 and c^s_{exp} is not valid or if the time component in the configurations do not respect the monotone condition. Otherwise c^s_{exp} is linked to c^s_1, making the tree grow by one node and one edge. The factor ε represents the incremental step taken during the search. Large steps make the trees grow quickly but with more difficulty in capturing the free configuration space around obstacles. Inversely, too small values generate roadmaps with too many nodes, slowing down the algorithms.

Path Smoothing. When a solution is found, a final step for smoothing the path is required. We use here the popular approach of applying several linearization steps. Each linearization consists of selecting two random configurations c^s_a and c^s_b along the solution path in C^S (not necessarily landmarks) and replacing the subpath between c^s_a and c^s_b by the straight interpolation between them, if the replacement is still a valid path. Note that the time component in c^s_a and c^s_b are as well interpolated and smoothed. The process is repeated until valid replacements are difficult to find or until a time threshold is reached. This simple process works well in practice and has both the effect of smoothing and shortening the path, which are obvious properties expected in natural motions.

4.2 Configuration Sampling

The sampling routine guides the whole search and is of extreme importance in determining the quality of a solution and how fast it is found. It is therefore important to define meaningful joint parameterizations, joint limits and search heuristics for both reducing the search space and guiding the search to more realistic postures. We pay particular attention here to the joints of the arm linkages due to their importance for object manipulation.

Joint Parameterization. The first step for ensuring anatomically plausible postures is to impose meaningful joint range limits on the articulations of the skeleton. For anatomical articulations with a 3 DOF rotation, e.g. the shoulder, we use the natural swing-and-twist decomposition [Grassia 1998]. The remaining joints are either parameterized with Euler angles or by a swing rotation.

For instance in the arm linkages the swing-and-twist decomposition is used to model the shoulder (3 DOFs). The elbow has flexion and twist rotations defined with two Euler angles (2 DOFs), and the wrist has a swing rotation (2 DOFs) parameterized exactly as a swing-and-twist, however considering the twist rotation to be always 0. The linkages of the legs are similarly parameterized.

Joint Limits. The swing parameterization allows the use of spherical polygons [Korein 1985] for restricting the swing motion. Spherical polygons can be manually edited for defining a precise bounding curve. However we follow a simpler, more efficient, and still acceptable solution for bounding swing limits based on spherical ellipses [Grassia 1998]. In this case, a swing rotation can be checked for validity simply by replacing the axis-angle parameters into the ellipse's equation. The twist and flexion rotations of the remaining DOFs are correctly limited by minimum and maximum angles.

Collision Detection. In order to achieve complex collision-free motions, we take into account the full geometries of the characters and the environment when checking for collisions. The VCollide package [Gottschalk et al. 1996] is employed for querying if body parts self-intersect or intersect with the environment.

Search Heuristics. We control the overall quality of the planned motions by properly adjusting sampling heuristics. Uniformly sampling valid postures has the effect of biasing the search toward the free spaces. For example, in several cases where the character manipulates objects, there are obstacles in front of the character and larger volumes of free space are located at the sides of the character. Although these are indeed valid areas, realistic manipulations are mainly carried out in the smaller free spaces in front of the character.

A simple correction technique for such cases consists of highly biasing the sampling towards the bent configuration of the elbow. This has the effect of avoiding solutions with the arm outstretched, resulting in more natural motions. As we perform a bidirectional search, it also contributes to decomposing the manipulation in two distinct phases: bringing the arm closer to the body and then extending it towards the goal. Our biasing method starts sampling the elbow flexion DOF with values in the interval between 100% and 90% of flexion, and as the number of iterations grow, the sampling interval gets larger until reaching the joint limits.

For other less important joints, e.g. the wrist or spine joints if used, the sampling is also similarly biased to a smaller range than their validity range, resulting in more pleasant postures as these joints are usually secondarily used for avoiding obstacles.

The sampling routine can be even interactively customized by choosing different values for the sampling intervals used for sampling each considered DOF. For example by adjusting the intervals of the shoulder DOFs we are able to control the overall location of the obtained arm motion. The top row of Figure 2 shows an example where the x-component of the shoulder swing was sampled between 50 and 100 degrees, generating only relatively low arm postures during the search. In the solution shown in the bottom row however, we choose to sample higher arm postures. Such example illustrates that we are able to control the overall quality of the motion and avoid repetitive results.

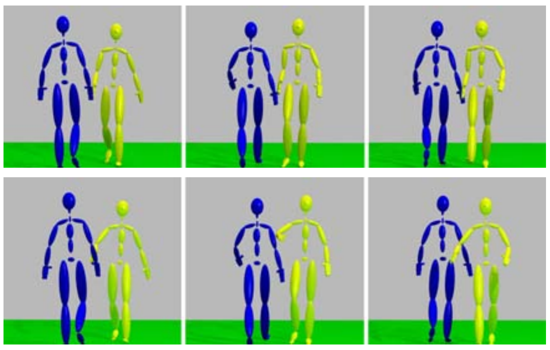
Figure 2: Two alternative solutions for correcting arm collisions obtained by choosing different sampling heuristics.

Final Sampling Routine. The final sampling routine can be summarized as follows:

1. Configuration $c^s_{rand} = (c^p_{rand}, t_{rand}) \in C^S$ is generated having the values in c^p_{rand} randomly sampled in the described parameterizations based on swings, twists and Euler angles; inside individually placed range limits and following the appropriate sampling heuristics. The time component t_{rand} is uniformly sampled in $[t_{init}, t_{goal}]$.

2. The state of the world is set with $w(t_{rand})$ and configuration $m^m(t_{rand})$ is applied to the character.

3. Configuration c^p_{rand} is applied to the character.

4. Finally the positions of all objects are updated and tested for collisions; if no collisions are found c^s_{rand} is returned as a successful valid configuration, otherwise the sampling routine returns to step 1.

5 Inverse Kinematics

Inverse Kinematics is an important component of our overall method. Although the planner does not require the use of IK during its execution, our IK allows us to easily (and interactively) specify goal arm and leg postures to be used as input to the planner. In particular for interactive grasping, the use of IK allows the user to define goal arm postures for the planner on-line, by simply selecting goal hand positions in the workspace.

In order to obtain realistic and fast results, we implemented an analytical IK formulation [Tolani and Badler 1996] that produces joint values directly in our arm and leg parameterizations with meaningful joint limits based on swings and twists. Note that for each arm or leg, there are 7 DOFs to be determined for reaching a given hand position and orientation goal. The problem is under-determined and the missing DOF is modeled as the *swivel angle*, which is an extra parameter specifying the desired rotation of the elbow (or knee) around the wrist-shoulder (or ankle-hip) axis.

We have furthermore integrated in the IK a simple search strategy that automatically searches for a swivel angle leading to a valid (and therefore collision-free) configuration. Equipped with such automatic posture search, the IK and the planner are able to produce complex collision-free animations for reaching given hand targets interactively.

We start solving the IK with the desired initial swivel angle, which is usually extracted from the current character posture. Then, the posture given by the IK solver is checked for validity. If the posture is not valid, the swivel angle is incremented and decremented by δ and the two new postures given by the IK solver are again checked for validity. If a valid posture is found the process successfully stops. Otherwise, if given minimum and maximum swivel angles are reached, failure is returned. Faster results are achieved in a greedy fashion, i.e. when Δ increases during the iterations. As the search range is small this simple process is very efficient and the whole process can be limited to a few number of tests. Note that both joint limits and collisions are avoided in an unified way.

6 Applications and Results

We have integrated the methods described in this paper in the DANCE animation system [Shapiro et al. 2005]. Multiple arms and leg targets can be specified and solved by our planner interactively. Targets can be dynamic and/or attached to any objects or body parts. Characters can be instructed to grab, drop and move objects. Several tasks can be specified simultaneously and synchronized with arbitrary keyframe motions applied to the characters.

In the remainder of this section we present several results obtained with our system. We group them by two key applications that demonstrate the versatility and the effectiveness of our approach. For a better presentation of the results we refer the reader to the accompanying video and our website (removed for anonymity).

6.1 Motion Correction

Our planner introduces an effective way to correct portions of motions that are found to produce collisions with new objects in the environment or with new objects attached to the character. Such situations are common when reusing motions in new environments or new characters. Our planner is able to search for an alternative motion for the problematic limb which is both valid and in synchronization with the original motion and any moving objects.

Let m be a given motion affecting the full configuration space C^F of the character. We want to correct a portion of m that was found to obtain collisions. For solving this kind of problem, we define times t_{init} and t_{goal} such that interval $[t_{init}, t_{goal}]$ spans the problematic period of the motion.

Let m be decomposed in two parts, such that $m(t) = (m^m(t), m^p(t)) \in C^M \times C^P$. The problem is then solved by planning a new path between $(m^p(t_{init}), t_{init})$ and $(m^p(t_{goal}), t_{goal})$ in C^S. If the planner is successful, the result will be a collision-free motion that is used to replace m^p during interval $[t_{init}, t_{goal}]$.

We present several examples in this paper. Figure 4 (a) presents a valid walking motion that becomes invalid when an umbrella is attached to the right hand of the character. The umbrella collides with the post in several frames of the sequence. Figure 4(b) presents the corrected motion after the planner is applied to produce a new synchronized motion for the joints of the right arm. The same walking motion was also successfully corrected by our planner in a new environment containing two posts (Figure 1). Other correction examples are shown in Figure 2 and Figure 4(c,d).

6.2 Interactive Object Manipulation

Object manipulation tasks for moving characters can be complex and computationally expensive to synthesize. Our planner generates realistic results of such highly complex tasks by synchronizing synthesized arm motions with locomotion sequences. For instance in Figure 4(e) we generate a motion where the character grasps a dynamic target through a moving ring while under the influence of an *idle* motion affecting its body. Figure 4(f) shows an even more complex motion where the character is asked to solve the same task but while transitioning from walking to running. Figure 4(g) shows a character walking and at the same time grabbing the hat of another walking character.

For this kind of problem we first specify hand targets on the objects to be grasped. Let $h = \{p, q\}$, $h \in \mathbb{R}^3 \times \mathbb{S}^3$, be a *hand target* described as a target position and orientation for the wrist joint of the character in global coordinates, to be reached at a given time t_b.

Let again $m(t) = (m^m(t), m^p(t)) \in C^M \times C^P$ be a motion as described in Section 6.1. We want now to modify motion m such that at time t_b the character wrist joint is located at the given hand target h, and as the modified motion has to be performed in a cluttered environment, it has to be collision-free.

We now determine times t_a and t_c such that $t_a < t_b < t_c$. Then the problem is solved in two steps: first a path in C^S is planned between $(m^p(t_a), t_a)$ and (c_h^p, t_b) and then a second path in C^S is planned between (c_h^p, t_b) and $(m^p(t_c), t_c)$. Configuration $c_h^p \in C^P$ is determined by employing our IK (Section 5), in order to determine the best arm configuration that reaches the hand target h.

The sequences in Figure 4(h-j) show several complex manipulation examples. The obtained results show realistic motions where planned arm manipulation sequences are perfectly synchronized with the walking motion. In this example, 10 planned sequences were used for synchronizing 5 different object manipulations: grasping a piece of cheese from inside a box, dropping it on the table, grasping a hat with the right hand, turning off the lights with the left hand and then placing the hat on the head.

We have also implemented a system to interactively instruct a character to reach for arbitrarily located objects, in synchronization with an on-line locomotion planner. We therefore compute the final motion in two steps: first a path is planned such that the character arrives close enough to the object to grasp with the hand. A motion captured sequence is then deformed to fit the computed path, and before the locomotion is finished, we compose a synchronized arm motion with the locomotion.

7 Discussion

One important characteristic of our method is the random nature of the planner. It ensures that the obtained motions are always different, greatly improving the realism in interactive applications of autonomous characters. At the same time, we are also able to control the overall aspect of the obtained results by choosing different body motions to synchronize with and search heuristics (Figure 2).

The performance of the planner greatly depends on the complexity of the environment. For instance in the complex scenario of Figure 4(h-j), the collision detection is handling 30K triangles and the planner took about 2 seconds to both compute and smooth each of the planned motions. In the simpler environments the performance is about two times faster.

Limitations and Extensions. Although our results are realistic, further processing could still be employed. For instance, dynamic filters could be applied for ensuring the balance of the character. However, this would penalize the overall performance of the method. We chose not to employ a more time-consuming method, such as those described by [Yamane et al. 2004] in the interests of speed. Our method currently serves as an interactive application whereby an animator can quickly edit and change the motion within seconds to his or her tastes.

Although the examples presented here show the planner is mainly applied to arms and legs, it can also be applied to any set of open linkages. It can be as well employed sequentially, for example for synchronizing the motions of several limbs: first, the motion of one limb is planned in synchronization with the given external motions, resulting in a new composite motion. Then, a second limb motion can be synchronized with the previously obtained motion. The process can be repeated until all limbs are planned and synchronized. The result achieved is a decoupled priority-based (due to the chosen order) planning process. Note that limbs may belong to different characters, as in the example shown in Figure 3.

The examples here could also use longer linkages on the same character, such as those that include the arm and torso to accommodate bending and twisting of the waist and trunk. The risk of using larger IK linkages is the deteriorating effect on the resulting realism that such a solution would provide. Since many IK solutions do not take into account physics or changes to the COM or momentum of the body, the longer the IK chain used, the less realistic the final motion will be. This could be overcome by either using an additional dynamic filtering as a postprocessing step, or employing an IK that accommodates changes to the rest of the body, such as shown by [Grochow et al. 2004].

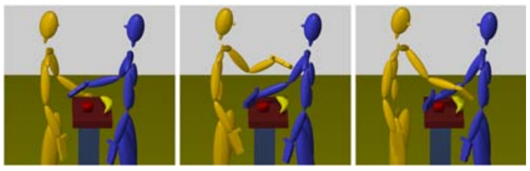

Figure 3: The motion of the character on the left side was planned after the motion of the character on the right side, achieving synchronized simultaneous graspings.

8 Conclusion

We have presented a new approach for motion editing based on planning motions in synchronization with pre-designed (or recorded) motion sequences and moving objects. In general, our method is able to solve arbitrary spatio-temporal constraints among obstacles and takes into account dynamic environments.

By relying on a hybrid approach, we are able to address the difficult constraints imposed by object manipulations, achieve realistic results and still leave space for designers to customize and personalize the underlying motion sequences.

9 Acknowledgements

The work in this paper was partially supported by NSF under contract CCF-0429983. We would also like to thank Intel Corp., Microsoft Corp., Alias Wavefront and ATI Corp. for their generous support through equipment and software grants.

References

ARIKAN, O., AND FORSYTH, D. A. 2002. Interactive motion generation from examples. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 483–490.

BANDI, S., AND THALMANN, D. 2000. Path finding for human motion in virtual environments. *Computational Geometry: Theory and Applications 15*, 1-3, 103–127.

CHOI, M. G., LEE, J., AND SHIN, S. Y. 2002. Planning biped locomotion using motion capture data and probabilistic roadmaps. *ACM Transactions on Graphics 22*, 2, 182–203.

GOTTSCHALK, S., LIN, M. C., AND MANOCHA, D. 1996. Obbtree: A hierarchical structure for rapid interference detection. *Computer Graphics SIGGRAPH'96 30*, Annual Conference Series, 171–180.

GRASSIA, S. 1998. Practical parameterization of rotations using the exponential map. *Journal of Graphics Tools 3*, 3, 29–48.

GROCHOW, K., MARTIN, S., HERTZMANN, A., AND POPOVI, Z., 2004. Style-based inverse kinematics.

HSU, D., KINDEL, R., LATOMBE, J., AND ROCK, S. 2002. Randomized kinodynamic motion planning with moving obstacles. *International Journal of Robotics Research 21*, 3, 233–255.

KALLMANN, M., AUBEL, A., ABACI, T., AND THALMANN, D. 2003. Planning collision-free reaching motions for interactive object manipulation and grasping. *Computer graphics Forum (Proceedings of Eurographics'03) 22*, 3 (September), 313–322.

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

KOREIN, J. U. 1985. *A Geometric Investigation of Reach*. The MIT Press.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 473–482.

KUFFNER, J. J., AND LATOMBE, J.-C. 2000. Interactive manipulation planning for animated characters. In *Proceedings of Pacific Graphics'00*. poster paper.

KUFFNER, J. J., AND LAVALLE, S. M. 2000. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings of IEEE Int'l Conference on Robotics and Automation (ICRA'00)*.

KWON, T., AND SHIN, S. Y. 2005. Motion modeling for on-line locomotion synthesis. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, New York, NY, USA, 29–38.

LAI, Y.-C., CHENNEY, S., AND FAN, S. 2005. Group motion graphs. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

LAU, M., AND KUFFNER, J. 2005. Behavior planning for character animation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

LAVALLE, S., AND KUFFNER, J., 2000. Rapidly-exploring random trees: Progress and prospects. In Workshop on the Algorithmic Foundations of Robotics.

LAVALLE, S. 1998. Rapidly-exploring random trees: A new tool for path planning. Tech. Rep. 98-11, Iowa State University, Computer Science Department, October.

LIU, Y., AND BADLER, N. I. 2003. Real-time reach planning for animated characters using hardware acceleration. In *Proceedings of Computer Animation and Social Agents (CASA'03)*, 86–93.

PETTRÉ, J., LAUMOND, J.-P., AND SIMEÓN, T. 2003. A 2-stages locomotion planner for digital actors. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 258–264.

SCHWARZER, F., SAHA, M., AND LATOMBE, J.-C. 2002. Exact collision checking of robot paths. In *Proceedings of the Workshop on Algorithmic Foundations of Robotics (WAFR'02)*.

SHAPIRO, A., FALOUTSOS, P., AND NG-THOW-HING, V. 2005. Dynamic animation and control environment. In *GI '05: Proceedings of the 2005 conference on Graphics interface*, Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 61–70.

TOLANI, D., AND BADLER, N. 1996. Real-time inverse kinematics of the human arm. *Presence 5*, 4, 393–401.

YAMANE, K., KUFFNER, J. J., AND HODGINS, J. K. 2004. Synthesizing animations of human manipulation tasks. *ACM Transactions on Graphics (Proceedings of SIGGRAPH'04) 23*, 3, 532–539.

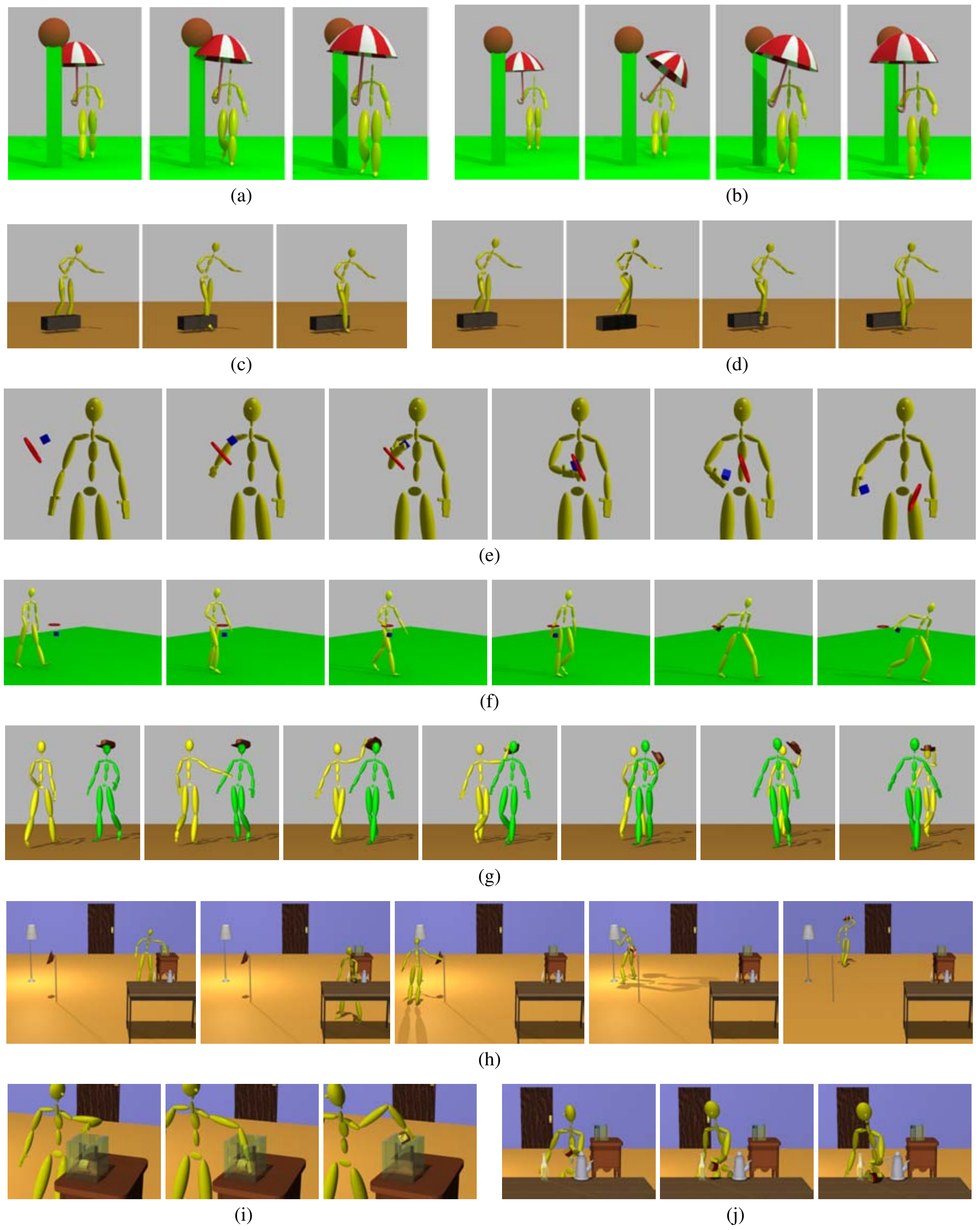

Figure 4: Sequences (a) and (c) have collisions and are corrected by our planner, which produced (b) and (d). Sequences (e) and (f) show examples of a moving cube being grasped from inside a moving ring. The character in sequence (g) steals the hat of another character while both are walking. Sequence (h) shows several object manipulations planned around obstacles and in synchronization with a long walking motion. Details of grabbing and dropping the cheese are shown in sequences (i) and (j).