Animation Planning for Virtual Characters Cooperation

CLAUDIA ESTEVES, GUSTAVO ARECHAVALETA, JULIEN PETTRÉ, and JEAN-PAUL LAUMOND
Laboratoire d'Analyse et d'Architecture des Systèmes, Toulouse, France

This paper presents an approach to automatically compute animations for virtual (human-like and robot) characters cooperating to move bulky objects in cluttered environments. The main challenge is to deal with 3D collision avoidance while preserving the believability of the agent's behaviors. To accomplish the coordinated task, a geometric and kinematic decoupling of the system is proposed. This decomposition enables us to plan a collision-free path for a reduced system, then to animate locomotion and grasping behaviors independently, and finally to automatically tune the animation to avoid residual collisions. These three steps are applied consecutively to synthesize an animation. The different techniques used, such as probabilistic path planning, locomotion controllers, inverse kinematics and path planning for closed kinematic chains are explained, and the way to integrate them into a single scheme is described.

Categories and Subject Descriptors: I.3.7 [**Computer Graphics**]: Three-dimensional Graphics and Realism; G.3 [**Probability and Statistics**]: Probabilistic Algorithms

General Terms: Algorithms

Additional Key Words and Phrases: Autonomous characters, behavior modeling, motion control, motion planning

1. INTRODUCTION

Over the last few decades, many attempts to animate the human figure have been made [Parent 2001; Earnshaw et al. 1998], each of them attaining a certain degree of autonomy, interactivity, and user-controllability.

The human figure has been frequently represented in computer animation in the same way articulated mechanisms are represented in robotics [Badler et al. 1993]. Nevertheless, the techniques developed to generate the motions for these mechanisms have been different in both areas. In computer animation, the interest has been mainly in generating realistic-looking motions, while in robotics the goal has been to automate motion generation regardless of its realism.

In recent years, applications arising in both areas (e.g., ergonomics, interactive video games, etc.) have motivated researchers to automatically generate human-like plausible motion. Consequently, this work deals with the development of an automatic motion strategy for the cooperation of two or more virtual characters that transport an object in a 3-dimensional cluttered environment. The virtual characters are considered to be either human figures with walking capabilities, referred to in this work as *mannequins*, or mobile robots. This work finds its applications mainly, but not exclusively, in PLM (Product Lifecycle Management) as in the maintenance and operation of industrial facilities [Badler et al. 2002].

C. Esteves and G. Arechavaleta benefit from SFERE-CONACyT grants. This work is partially funded by the European Community Projects FP5 IST 2001-39250 Movie and FP6 IST 002020 Cogniron.
Authors' addresses: {cesteves,garechav,jpettre,jpl}@laas.fr.

In this context, several behaviors should be combined within a single animation sequence: agents should walk or slide while manipulating a bulky object coordinately with the other characters. Here, the main challenge is to achieve 3D collision avoidance while preserving the believability of the agents behaviors.

From an algorithmic perspective, our approach is a centralized one. Indeed, we show how to model the global task within a single system which contains all the degrees of freedom (DOFs) of the agents and the object. This system is automatically built by computing a *reachable cooperative space*. Then, three consecutive steps are performed.

(1) Plan a collision-free trajectory for a reduced model of the system.
(2) Animate, in parallel, locomotion and manipulation behaviors.
(3) Tune the generated motions to avoid residual collisions.

These steps are applied by making use of a probabilistic motion planner to compute the collision-free trajectories; motion controllers adapted to each kind of character for both locomotion and grasping behaviors; and path planning algorithms for closed kinematic chains to deal with coordinated behaviors.

The remainder of this article is structured as follows. Section 2 gives a brief overview of related work. Section 3 introduces the different techniques used in this work. In Section 4, our system is described and the underlying principles of our approach are stated. Section 5 details the three steps performed in order to generate an animation. Experimental results are shown and discussed in Section 6.

2. RELATED WORK AND CONTRIBUTION

In order to build our motion planner, we have chosen the techniques that best provide the desired characteristics to the resulting animation; that is, believable and automatic motion, precise manipulation and combined behaviors.

Our system is represented with a tree-like structure with a high number of degrees of freedom and kinematic constraints. This representation is well suited for probabilistic motion planning techniques. Several works have proposed motion planners based on this kind of probabilistic approach to produce collision-free human-like trajectories. However, they have mainly focused on generating trajectories either for locomotion or for manipulation. Our work addresses motion planning in the context of manipulating and walking at the same time.

2.1 Motion Planning for Walking Characters

In order to generate plausible human-like motions, techniques based on motion capture have frequently been used. These techniques have proven to be suitable for real-time motion generation. However, the examples in a motion library are limited in number, and it is often necessary to modify them to avoid repetitive motions. This is usually done by using interpolation procedures such as in Witkin and Popovic [1995]; Unuma et al. [1995]; Rose et al. [1998]. Because we are dealing with eye-believable locomotion generation which is repetitive by nature, controllers based on motion capture are particularly adequate.

Motion planners exploiting these kind of techniques have been described in the literature. In Kuffner [1998] and Choi et al. [2003], two-step motion planners for walking virtual mannequins are presented. The first approach consists in first planning a collision-free path for a cylinder in a two-dimensional world and then following it by means of a PD controller that uses cyclic motion capture data. The latter approach is capable of dealing with rough terrains by planning the mannequin's footprints and then applying and adapting captured motion to attain the planned footprints. Our work is based on the first approach, extended in Pettré et al. [2003]. Here, the 3-dimensionality of the environment is taken into account by applying a collision avoidance-based warping method described in Section 5.3.

2.2 Motion Planning for Object Manipulation

As far as manipulation and reach planning for virtual mannequins are concerned, the problem has been tackled using different approaches.

In Liu and Badler [2003], the authors plan and synthesize collision-free reaching motions for 7-DOF arms in real time by using graphics hardware to detect collisions. Their planning framework reasons directly in the workspace. However, this kind of planning approach has not proven to be more efficient than the configuration-space planning approaches used in Kallman et al. [2003] and Yamane et al. [2004].

In Kallman et al. [2003], the authors plan reaching motions for a 22-DOF system (9 DOFs for each arm and 4 additional DOFs to specify the mannequin's body posture). Inverse kinematics algorithms are used to specify the initial and final configurations, and a roadmap is constructed based on the cost computed for each configuration. This approach can manage obstacle displacements by dynamically updating the roadmap.

In Koga et al. [1994] and Yamane et al. [2004], a trajectory is first found for the object to be manipulated or grasped and then the loop is closed between the arms and the object to follow the computed trajectory. Our work is conducted in the same spirit. Our contribution here is to address the grasping task at the planning level itself.

2.3 Combining Behaviors

The problem of combining behaviors of articulated human figures has been frequently tackled by applying behavior-based controllers as in Perlin [1995]; Blumberg and Galyean [1995]; Brand and Hertzmann [2000]. Here, motion capture data is labeled as containing one particular behavior or characteristic (run, walk, scratch head, etc.). A new walking-scratching-head sequence can be generated by interpolating configurations of the original captured data.

Kovar et al. [2002] captures a set of labeled motion examples in a graph. A believable animation is produced by composing motion captures corresponding to chosen edges and the transitions between them (nodes). A path can be followed by minimizing the error between the path in the graph and the user-sketched one. In Kallmann et al. [2004], a sampling approach using parameterized motion primitives is used to satisfy a given task. Motion primitives are chosen by means of a tree in which nodes are valid configurations reachable by more than one primitive and edges represent the paths between them. Their chosen parameterization allows the computation of trajectories maintaining constraints such as balance along it.

Physically-based solutions are described in Shiller et al. [2001] where a simple motion planner is presented to combine behaviors (walking, crawling and side-walking) in a sequential manner. Another approach is presented in Faloutsos et al. [2001] where individual controllers generating different behaviors are managed by a supervisor controller.

In our work, behaviors are combined by using the geometric and kinematic decoupling approach described in Section 4 which consists of the decomposition of the DOFs of the character. In this work, we combine both grasping and locomotion behaviors in a continuous way.

Therefore, the main contribution of this article is to address all these issues in a single scheme and, at the same time, deal with three-dimensional collision avoidance.

3. TECHNIQUES OVERVIEW

In order to generate complete motion sequences of one or more virtual characters transporting a bulky object in cluttered environments, we use three main components:

—a motion planner that handles open and closed kinematic chains,

—motion controllers adapted for virtual mannequins and for virtual robots,

—a three-dimensional collision avoidance editing strategy.

Many existing techniques can be used to cover these requirements. In the paragraphs that follow, we describe those that we experienced as the best adapted to our problem.

3.1 Probabilistic Motion Planning Techniques

3.1.1 *Probabilistic Roadmaps.* The aim of this method is to obtain a representation of the topology of the collision-free space [Latombe 1991] in a compact data structure called a *roadmap*. Such a structure is used to find collision-free trajectories and is computed without requiring an explicit representation of the obstacles in the configuration space. A roadmap can be obtained by using two types of algorithm: *sampling* or *diffusion*.

The main idea of the sampling technique (introduced as *PRM* in Kavraki et al. [1996]) is to draw random configurations lying in the free space and to trace edges to connect them with neighbor samples. Edges or local paths should also be collision-free and their form depends on the kinematic constraints of the moving device.

The principle of diffusion techniques [LaValle 1998] consists of sampling the collision-free space with only a few configurations called *roots* and to diffuse the exploration in the neighborhood to randomly chosen directions.

In this work, we use a variant of the first approach, the *Visibility PRM* [Siméon et al. 2000]. Here, there are two types of nodes: the guards and the connections. Nodes are added if they are not visible from previously sampled nodes (guard nodes) or if they allow two or more connected components of the roadmap (connection nodes) to be linked. The generated roadmap is more compact than the one obtained using PRM alone.

3.1.2 *Planning for Open Kinematic Chains.* When using sampling-based methods, a trajectory is found in a roadmap by using a two-step algorithm consisting of a learning phase and a query phase. For an articulated mechanism, the roadmap is computed in the learning phase by generating random configurations within the range allowed for each DOF. In the query phase, the initial and final configurations are added as new nodes in the roadmap and connected with the chosen steering method. Then, a graph search is performed to find a collision-free path between the start and goal configurations. If a path is found, then it is smoothened to remove useless detours. Afterwards, it is converted into a trajectory (a time-parameterized path) by means of classical techniques (e.g., Lamiraux and Laumond [1997]).

The advantage of using sampling methods when dealing with complex static environments as in this work, is that the learning phase can be precomputed offline and then the roadmap is used online to find a trajectory, saving time on each query.

3.1.3 *Planning for Closed Kinematic Chains.* In order to handle the motions of closed kinematic mechanisms, some path planning methods have been proposed in the literature [LaValle et al. 1999; Han and Amato 2000; Cortés and Siméon 2003]. In our work, we have chosen to use the *Random Loop Generator (RLG)* proposed in Cortés and Siméon [2003] and summarized in Algorithm 1. In this method, a closed kinematic chain is decomposed into active and passive parts. The main idea is to progressively decrease the complexity of the closed kinematic chain processed at each iteration until only the configuration of the passive part of the chain remains to be solved by means of an inverse kinematics algorithm.

Algorithm 1. RLG_SingleLoop

```
Input : the loop $L$
Output : the configurations $q[n_{sol}]$
begin
    $q^a \leftarrow$ Sample_$q^a(L)$;
    $q^p[n_{sol}] \leftarrow$ Compute_$q^p(L, q^a)$;
    if $n_{sol} = 0$ then return Failure;
    else $q[n_{sol}] \leftarrow$ CompoundConf$(L, q^a, q^p[n_{sol}])$;
end
```

The joint values of the active chain q^a are computed sequentially by using Algorithm 2 where J_b through J_e are the active joints on the chain. The range at which each joint is sampled depends on the configuration of the previously processed joints. This closure range for each joint is approximated by a simple bounding volume whose parameters depend on the geometry of the chain. The *reachable workspace* of a kinematic chain is defined as the volume which the end-effector can reach. The reachable workspace is automatically approximated by a spherical shell, defined as the intersection of the volume between two concentric spheres and a cone. The parameters to construct this volume are mainly the origin of the chain and the maximum and minimum extensions of the chain. A guided random sampling of the configuration of the active part is done inside this volume by limiting the value for each joint to the interval computed in the function ComputeClosureRange in Algorithm 2. In function CompoundConf, for a sampled value of q^a, the value of q^p is computed by solving an inverse kinematics problem. When several loops are present in the mechanism, they are processed as separate closed chains.

When the roadmap is constructed, a trajectory is found in the same way as for open kinematic chains.

Algorithm 2. Sample_q^a

```
Input : the loop $L$
Output : the parameters $q^a$
begin
    $(J_b, J_e) \leftarrow$ InitSampler$(L)$;
    while not EndActiveChain$(L, J_b)$ do
        $I_c \leftarrow$ ComputeClosureRange$(L, J_b, J_e)$;
        if $I_c = \emptyset$ then goto line 2;
        SetJointValue$(J_b$, Random$(I_c))$;
        $J_b \leftarrow$ NextJoint$(L, J_b)$;
        if not EndActiveChain$(L, J_e)$Switch$(J_b, J_e)$;
    end
end
```

3.2 Motion Generation Techniques

Once a trajectory is found, the appropriate motions to follow it as accurately as possible should be produced. In the following paragraphs, we describe the techniques used to generate such motions for walking as well as for manipulating.

3.2.1 *Manipulation Control.* Kinematics-based techniques allow the motion to be specified independently of the underlying forces that produced them. Motion can either be defined by specifying the value of each joint (forward-kinematics) or it can be derived from a given end-effector configuration (inverse-kinematics). These kinds of techniques have been frequently used to generate the motions of virtual mannequins as in Zhao and Badler [1994]; Yamane and Nakamura [2003]; Baerlocher and Boulic [2004]. In our approach, IK is used to provide precise control of the grasping behavior. Several IK algorithms for 7-DOF anthropomorphic limbs have been developed based on comfort criteria in

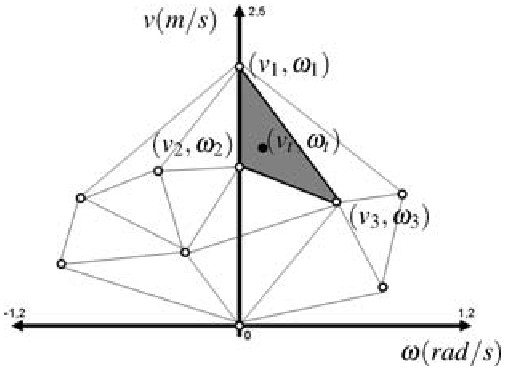

Fig. 1. Each captured source is represented as a point in the v–ω space according to their average linear and angular velocities. A new walking cycle will be obtained by interpolating its three closest sources.

order to best reproduce human-arm motions (e.g. Kondo [1991; Tolani et al. [2000]). We have chosen to use the analytic IK method presented in Tolani et al. [2000] to specify the configuration of our virtual mannequin's arms. To specify the configuration of virtual robot's 6-DOF arms, we use the algorithm described in Renaud [2000].

3.2.2 *Locomotion Control.* In order to specify the configuration of the walking virtual mannequin, we use the motion capture-based controller described in Pettré and Laumond [2005]. Here, the problem is modeled in such a way that the best motion examples to be blended as well as their respective weights are chosen by a simple geometric computation. The key idea is to represent all motion examples in a given library as single points lying in a 2-dimensional velocity space (linear and angular velocities) as in Figure 1. These points are then structured into a Delaunay triangulation, a well-known data structure in computational geometry, which allows efficient queries for point locations and nearest neighbor computations. The control scheme is based on a blending operator that combines three sources of motion capture. Their respective weights are automatically computed by solving a simple linear system with three unknown variables. The input parameters of the locomotion controller are, therefore, the sequence of sampled (v_t, ω_t), where v_t and ω_t are the linear and angular velocities of the virtual characters trajectory. Then, for each pair (v_t, ω_t) that represents one locomotion cycle, the three motion capture sources (v_1, ω_1), (v_2, ω_2), and (v_3, ω_3), with the closest average speeds to the sampled velocities are blended. As a result, a new locomotion cycle with the desired velocities is computed.

Finally, in order to obtain a continuous animation along the trajectory, the frequency characteristics of each cycle are computed and cycles are interpolated as described in Pettré and Laumond [2005].

4. MODELING THE SYSTEM

4.1 Virtual Characters

Our system is composed of two or more virtual characters and a movable object lying in a 3-dimensional cluttered environment. Each virtual character is classically represented as a hierarchy of rigid links connected by joints. In this work, we consider two kinds of virtual characters, human figures or mannequins and mobile robot manipulators.

The skeleton of our virtual mannequins is composed of 20 rigid bodies articulated by 18 joints with 53 degrees of freedom (DOFs). These joints and bodies are arranged in five kinematic chains that converge in the mannequin's root located on its pelvis (see Figure 2(a)).

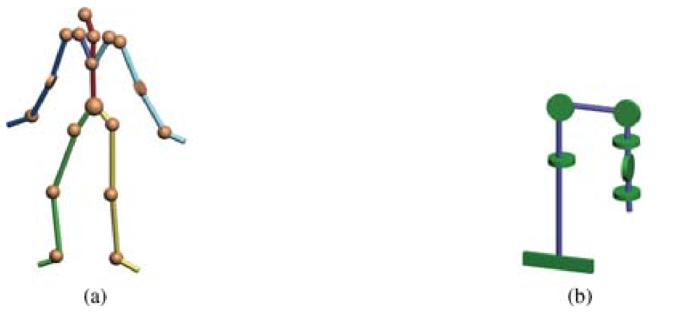

Fig. 2. Our virtual characters are represented as tree-like structures formed of rigid bodies linked with spherical, rotational, and planar joints.

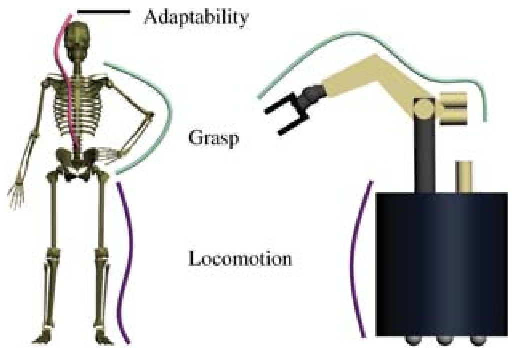

Fig. 3. Our system degrees of freedom are decomposed in three groups, locomotion, grasp, and adaptability.

Analogously, our virtual robot manipulators are composed of 7 rigid bodies linked by 6 joints and a mobile platform giving rise to a model with 9 degrees of freedom. These joints form one kinematic chain attached to the base of the robot (Figure 2(b)).

On top of the geometric model, kinematic constraints are imposed. For instance, we might want to consider that virtual human mannequins are allowed to walk only forwards or mobile manipulators to be differential drive robots. Kinematic constraints can also be imposed upon the object depending on the application, for example, keeping a tray with wine glasses horizontal. These constraints will be processed within the motion planning strategy described in Section 5.

4.2 Behavior-Based Kinematic Model

The main underlying principle of our work is a geometric decoupling of the system. This means that the system DOFs are decomposed in groups according to the main task they perform. The system contains the three groups of DOFs illustrated in Figure 3, locomotion, grasp, and adaptability.

Locomotion DOFs are the ones involved mainly in the steering of the character in the environment. For the virtual mannequin, these are the DOFs located in its legs and pelvis. For virtual robots, locomotion DOFs are the ones moving its base.

In a similar way, *Grasp* DOFs are in charge of the tasks that involve manipulation, that is, the arms of the characters. In this work, the mannequin's arms are redundant 7-DOF kinematic chains, and the virtual robots are equipped with 6-DOF nonredundant manipulators.

Adaptability DOFs are those involved in neither locomotion nor in manipulation but that allow a complementary posture control. For virtual mannequins, these DOFs lie in the head and spine. In our current virtual robots, there are no adaptability DOFs.

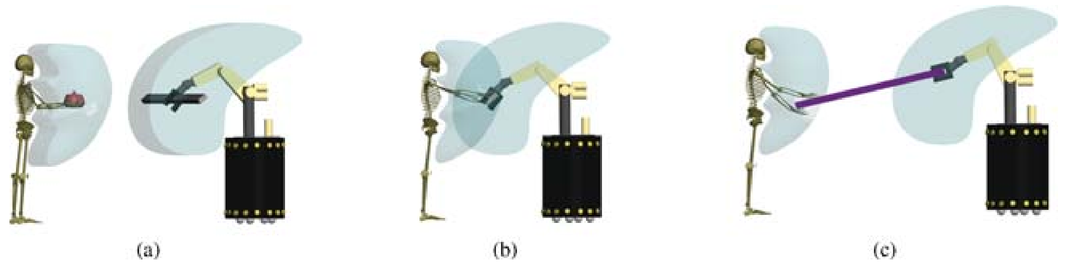

Fig. 4. (a) Individual reachable workspaces (b) To obtain a cooperative space for manipulation, the individual workspaces are intersected. (b) When manipulating a large object, the individual workspaces are enlarged by the objects size to compute the cooperative reachable workspace.

The advantage of such a geometric decoupling approach is that a reduced model of the system is obtained for each of the different steps of the planner. In this way, the control and description of the current task is simplified.

4.3 Reachable Cooperative Space

To attain cooperation between characters, a description of the space where the object can be manipulated is needed.

For a single virtual character manipulating an object with both hands, the space in which the object has to lie in order to remain reachable is defined by the possible solutions of the inverse kinematics algorithm applied to the arms. In this way, the reachable space can be approximated as described in Section 3.1.3 with the *spherical shells* technique consisting of the intersection of simple geometric shapes with the limits given by the maximal and minimal arm extension. Figure 4(a) shows an approximation of such a space for each type of character.

In this work, as the characters are supposed to carry the same object, we consider the *reachable cooperative space* as the intersection of all individual spaces (see Figure 4(b)). In order to achieve this intersection, a nonrigid link between the characters is considered.

Note that in the case of a large object, as in Figure 4(c), individual reachable spaces are not intersecting, nevertheless both characters are still holding the object. This is achieved by considering the object as the end effector of the arms kinematic chain for each of the characters. The individual space is therefore enlarged and the cooperative workspace obtained.

5. THE THREE-STEP ALGORITHM

Our approach relies mainly on three stages: planning, animating, and tuning. Several techniques are suitable for solving each stage. We have adopted some of them in a hierarchical manner to finally achieve collision-free planning motions for the whole system.

The user-specified inputs for the algorithm are:

— a geometric and kinematic description of the system A,
— geometric description of the environment E,
— maximal linear velocity and acceleration P,
— number of the desired frame-rate in the animation also stored in P,
— a motion library containing captured data from different walking sequences *MLib*,
— the maximum number of failures before the algorithm stops *ntry*,
— an initial and a final configuration.

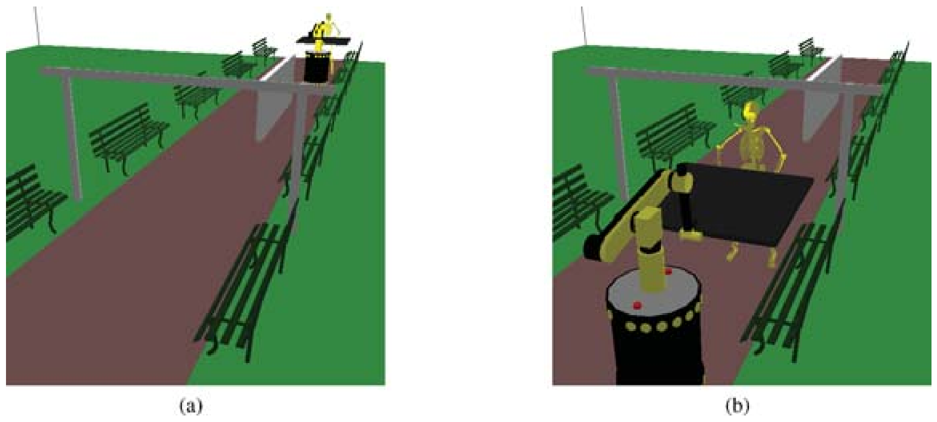

Fig. 5. Starting and goal configurations of the system for our workout example.

Algorithm 3. GLOBAL PLANNING

```
Input : the system $A$, the environment $E$, the list of parameters $P$
Output : the configurations $A(q_i)$
begin
    $stop \leftarrow$ False;
    while $\neg stop$ or $j < ntry$ do
        $path \leftarrow$ COMPUTEPATH($A_{simp}$, $E$);
        if $path = \emptyset$ then
            $stop \leftarrow$ True;
        end
        else
            $traj \leftarrow$ SAMPLING($path$, $P$);
            $A(q_i) \leftarrow$ GENERATEANIMATION ($traj$, $A$, $MLib$);
            if $\neg$ TUNING($A(q_i)$) then $j \leftarrow j + 1$;
            else
                $stop \leftarrow$ True;
            end
        end
    end
end
```

The output is an animated sequence of the combined behaviors $A(q_i)$. Algorithm 3 summarizes the various steps. In the next paragraphs, each stage is described. We use the workout example of Figure 5 to illustrate the algorithm.

5.1 Path Planning

In the planning phase, a reduced geometric model of the system is used. This simplified model is defined by three different elements: two boxes bounding the locomotion DOFs of each character and the object DOFs (see Figure 6). This means that a 12-DOF system is considered at this level. Six of them are the 3-dimensional position and orientation of the object. The other six are the planar position and orientation of each character's box. Three DOFs will be added to the reduced model for each additional character present in the scene.

Given the user-defined initial and final configurations of the system in the 3D environment (Figures 5(a) and (b), respectively) , the first procedure, COMPUTEPATH, plans a path for the simplified

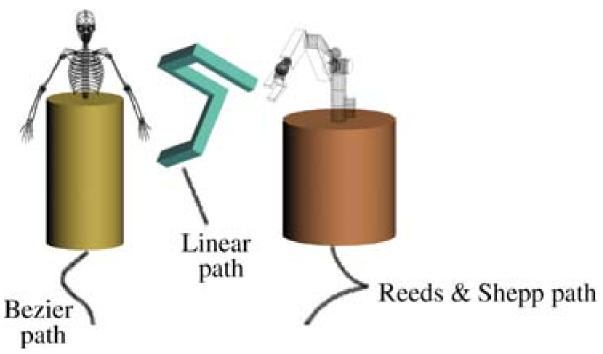

Fig. 6. A 12-DOF reduced geometric model of the system is employed in the planning phase. Each of the bodies will have an associated steering method while planning a path for the system.

model previously described. For this, the probabilistic roadmap method from Section 3.1.1 is used. This approach is performed in two steps, a learning and a query phase. The principle of the learning phase is to generate a random valid configuration for the system within the allowed range of the articulation limits for each DOF. In addition, in this stage, the cooperative reachable space as well as the maximal and minimal extension of the nonrigid link between the characters is computed. The random configurations are sampled within these limits.

During the sampling, a local planner tries to connect pairs of random configurations to incrementally construct a roadmap that captures the topology of the configuration space. Having the precomputed roadmap, the query phase performs a graph search to find feasible paths between the initial and final configurations.

The sampling strategy that we use is described in Section 3.1.1. This strategy captures well the topology of the configuration space in a compact roadmap. Local paths are computed by applying the adequate steering method. The selection of the steering method depends on the kinematic structure of each mobile entity that is part of the system. Let us consider, for instance, the construction of the local path for the virtual mannequin in the 12-DOF reduced model of Figure 6. Here, we chose Bezier curves of the third degree to approximate human-like trajectories in smooth curves that can be easily parameterized with four control points. Two of these points are the initial and final configurations of the mannequin's root and the mid-points are initial and final configurations translated by a user-defined distance in the initial or final root's direction. A Bezier curve is computed to connect each pair of intermediate configurations in the roadmap, therefore a path is a sequence of Bezier curves that share extremal control points. For the robot, we use Reeds and Shepp curves [Reeds and Shepp 1990] that account for the nonholonomic constraints (rolling without sliding) [Laumond 1998]. The object to be manipulated moves following straight line segments in its 6-dimensional configuration space. Note that no simplification is done for the model of the object to be carried. Its shape may be as complicated as desired, that is, there is no bounding box approximation for the object.

After the roadmap is constructed, a connecting path between the initial and final configurations is searched for. If the path is found, then the function SAMPLING transforms it into a trajectory (i.e., a time parametrized path) with user-defined velocity and acceleration constraints. It is important to note that at this level only the bodies attached to the locomotion and to the object DOFs are guaranteed to be free of collisions.

Figure 7 shows an example in the 3-dimensional environment. Here, the system is composed of a virtual mannequin, a mobile manipulator, and an object. The barrier in the middle of the walk path forces the robot to take the left side, while the human can still walk on the right side (Figure 7(a)). In

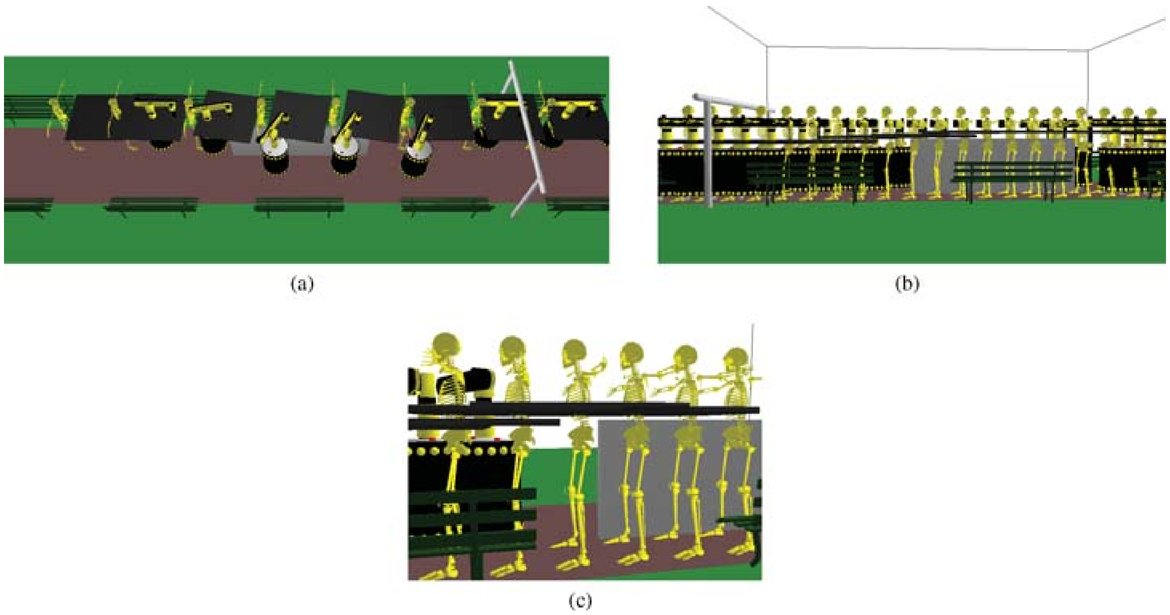

Fig. 7. (a) In the planning step, a trajectory is found for the 12-DOF system in a 3-dimensional environment. Here, the barrier in the middle of the scene causes the mannequins to take different paths. (b) Side view of the trajectory performed by the system. At this stage, locomotion and grasping behaviors are not yet synthesized. (c) The object is raised in order to avoid the barrier.

Figure 7, some of the configurations of the 12-DOF system are illustrated. It can be seen how the object is raised while traversing the barrier and lowered once it finishes. In this case, each of the characters follow the trajectory computed for their respective bounding cylinders according to their own steering methods. Note that the object remains within the reachable space between the virtual mannequin for the duration of the animation.

5.2 Behavior Control

At this stage, the trajectory for the 12-DOF simplified model is already planned. The next step is to synthesize the motions for the complete system involving the locomotion, adaptability, and grasp DOFs. This is done in the function GENERATEANIMATION by applying two different techniques: a locomotion controller to animate locomotion DOFs (pelvis and legs) and adaptability DOFs (spine and head) and an inverse kinematics algorithm adapted for each kinematic chain labeled as grasp DOFs (the mannequin's and the robot's arms).

The locomotion controller we have adopted is based on motion capture blending techniques. The result of this controller is a walking sequence for the virtual mannequin (see Section 3.2.2).

In order to synthesize the coordinated manipulation motions between the virtual characters, the IK solution for each arm is computed in order to reach the values imposed by the object configurations in the first stage (see Section 3.2.1). After applying the steps mentioned, a closed-chain mechanism is formed. As it is shown in Figure 8, two closed-loops exist. One is formed by the virtual mannequin and the object (body-arms-object) and the other by both characters (body-object-robot-floor).

Bearing in mind that the adaptability, as well as the grasp DOFs of each character could be in a collision state, a postprocessing stage is performed. In such a tuning phase, closure constraints are considered while the believability of the motions is preserved.

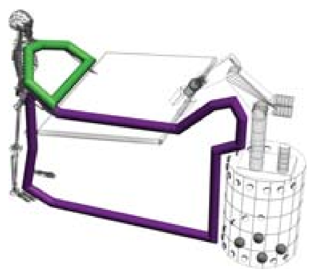

Fig. 8. In the animation phase, two closed kinematic chains are formed between the characters and the object.

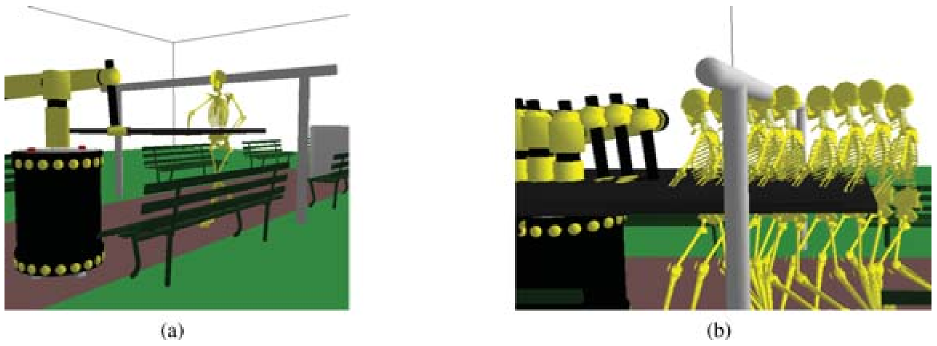

Fig. 9. (a) A configuration of the synthesized grasping and locomotion behaviors. (b) After the animation stage is performed, there can still be configurations where the upper body collides with the environment.

Figure 9(a) illustrates a configuration of the synthesized locomotion and grasping behaviors. Note that the character's head in Figure 9(b) still collides with the upper bar.

5.3 Automated Tuning

The purpose of this stage is to solve the possible residual collisions along the animated sequence. This is done by a local kinematic deformation of either the adaptability (spine-head) or grasp (arm-object) kinematic chains until a random collision-free configuration is reached. First, contiguous portions of the animation with collision in the same kinematic chain are identified and grouped into blocks. A new random configuration is generated for the colliding chain. Once the motion is modified there are three possible scenarios.

(1) The new motion contains a colliding configuration on the same kinematic chain. The random configuration is rejected and a new one is drawn.
(2) A collision for another kinematic chain is detected. In this case, the random configuration is stored as a solution for this chain.
(3) There is no collision detected. The configuration is kept and a new block is processed.

The collision-free motion generated randomly is usually not believable because of its high amplitude. To avoid this, a last step is required to progressively move the character away from the obstacle. Here we apply a warping procedure considering the two blocks, the original and the modified one. Such a procedure is typical in computer graphics to modify a sequence of key-frames when the two blocks have the same number of key-frames. The warping procedure consists in interpolating only the DOFs of the colliding chain. The parameters of the interpolation are controlled by the collision checker in

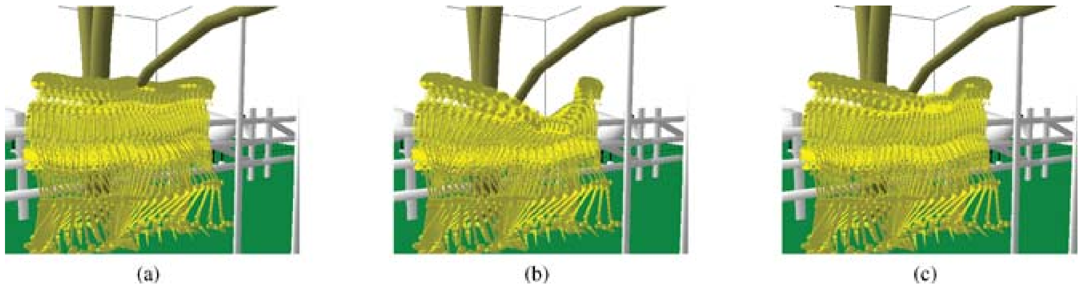
Fig. 10. (a) After synthesizing locomotion, a collision is found in the mannequin's head with the tree branch. (b) The collision is solved by finding a random collision-free configuration within the joint limits and warped with the previous animation. (c) The solution is optimized to obtain a smooth motion.

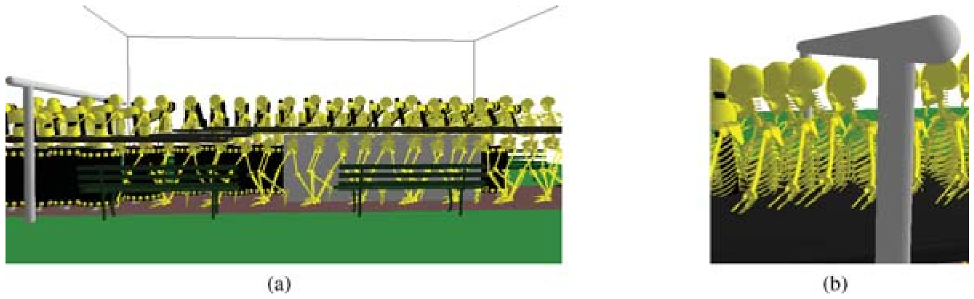
Fig. 11. (a) Some configurations of the resulting animation without residual collisions. (b) The collision with the upper bar is avoided in the tuning step.

order to provide a new configuration for the kinematic chain which is as close as possible to the original configuration while being free of collisions. This procedure is computed in function TUNING. Figure 10(a) shows a sequence where the mannequin's head collides with a tree branch. The colliding configurations are identified and a new, collision-free, solution is found as shown in Figure 10(b). Finally, the motion is optimized to preserve the eye-believability of the final animation as shown in Figure 10(c). In our barrier example, the tuning procedure was automatically applied at the end of the sequence of Figure 11(a), where the mannequin lowers its head in order to avoid the upper bar. This method has proven to work well when small deformations are needed.

If we consider the case of collisions involving the grasp DOFs, a local planner based on closed kinematic chains is used. In order to deal with multiloop closure constraints, we use the variant of RLG algorithm for multiple robots (see Section 3.1.3). For this, we consider the object DOFs as the active part of the closed kinematic chain, and the grasp DOFs as the passive part. The goal is to make the active chain reachable by all passive chains simultaneously. This is done by performing a guided-random sampling of the active chain within the intersection space formed between the reachable space of each passive chain. The configurations of passive chains are found using inverse kinematics. This procedure is illustrated in the pizza delivery example in Section 6.

5.4 Failure Recovery

After these three stages, if all configurations are not collision-free, the path generated in the first planning stage is invalidated. We remove from the roadmap the edge corresponding to the local path where the tuning step failed. Then a new global search is performed in order to find a new path.

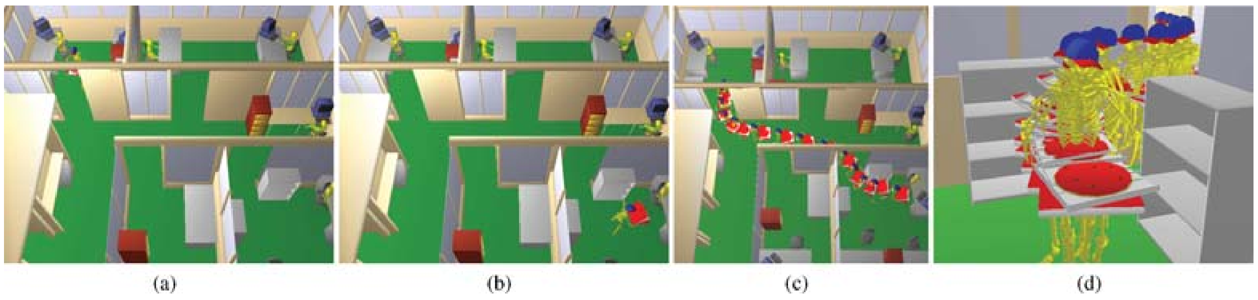

Fig. 12. (a) Initial configuration. (b) Final configuration. (c) Some configurations of the computed trajectory. (d) The mannequin moves his elbow in order to avoid collision with the bookshelf.

6. EXPERIMENTAL RESULTS

Our algorithm has been implemented within the motion planning platform Move3D [Siméon et al. 2001]. In the following paragraphs, examples of various scenarios and characters are presented.

6.1 Pizza Delivery Service

In this example, the virtual pizza delivery boy has to take a pizza box from one office to another. Here, we would like the mannequin to keep the boxes horizontal along the trajectory to prevent the pizzas from losing their toppings. For this, we impose kinematic constraints by restricting two of the six DOF of the free-flying object. This means that, in roll-pitch-yaw angle representation, we have removed the DOF allowing the object to pitch and roll.

Once the initial and final configurations (Figures 12(a) and (b)), velocity and acceleration constraints, and the number of frames are specified by the user, the algorithm is applied and the animation generated. In Figure 12(c), the resulting trajectory is shown as a sequence of configurations selected for image clarity.

After the locomotion and grasping behaviors were generated, a residual collision was found between the mannequin's arm and the bookshelf at the second office entrance. In this case, a solution was quickly found by modifying the elbow's configuration as shown in Figure 12(d). Here, this collision was not likely to be avoided in the original path because the doorways are narrow and the bookshelf is very near the final configuration.

In these examples, only four motion capture examples were used to synthesize locomotion. These capture examples included two straight lines at different speeds and two 45 degree turns, both left and right. This lack becomes evident when sharp turns are present in the path. This effect is apparent when the character turns and his feet slide on the floor. This can be corrected using more captured sources but it is interesting to note the good performance of the locomotion controller even with a low number of motion captured examples.

6.2 In the Factory

In this example, a virtual mannequin and a robot character have to transport a slab in a typical industrial environment with plenty of complex obstacles (pipes, drums, beams, ventilation units, etc.). Figure 13 shows some configurations of the resulting animation illustrating the trajectory followed by the system. As the system approaches the final configuration, the slab is turned as a result of the planning step in order to avoid the pipes.

In the animation stage, the locomotion behavior is animated and inverse kinematics is applied in order to grab the object in its new configuration. At the beginning of the trajectory, the human mannequin

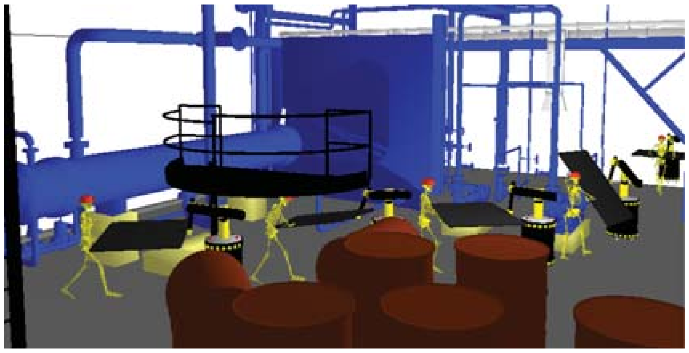

Fig. 13. The characters deal with several obstacles while cooperatively transporting a slab.

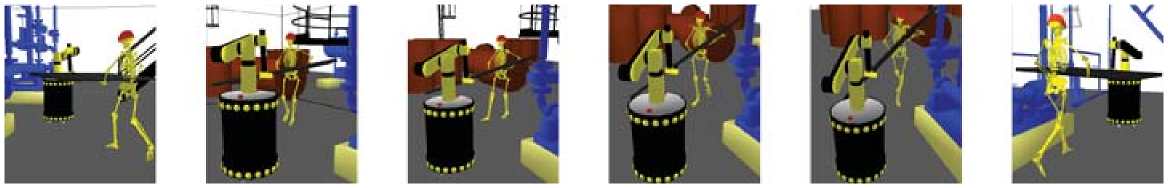

Fig. 14. Some configurations extracted from the final animation.

head collides with the balcony. This collision is automatically solved in the tuning step by bending the spine and head of the mannequin. It should be noted that, if the balcony were lower, it would be very difficult, if not impossible, to find a bending collision-free configuration for the mannequin. This is due to the fact that the tuning stage is only applied to the upper part of the body and, in order to avoid a lower balcony, a knee flexion would be required. Once the obstacle is left behind, the mannequin smoothly regains its posture to continue the cooperative task until the final configuration is reached.

Figure 14 shows a set of frames extracted from the final animation. Here, we can see that the motion of the system remains plausible after applying the three steps in the algorithm. Note that the trajectory planned for each of the agents in the scene remains collision-free and their position close enough to keep holding the slab.

6.3 Buren's Columns

This example in our version of the Buren's Columns involves two virtual mannequins that handle a large plate. Here, we have introduced kinematic constraints on the plate in order to keep it horizontal. Because of this, a 10-DOF reduced model is considered in the planning step. Figure 15 shows a view of the complexity of the environment and the system.

Figure 16 illustrates some frames from the resulting animation. The virtual mannequins walk along the computed trajectory carrying their plate. The object configuration is then modified because of the increasing height of the columns. In the third frame, an upper view of the system shows that the computed trajectory for the plate is collision-free. The mannequins continue to raise their object until it is safe to put it down. The final two frames show that one of the agents collides with the bar. This collision is solved in the tuning step by bending the mannequins spine.

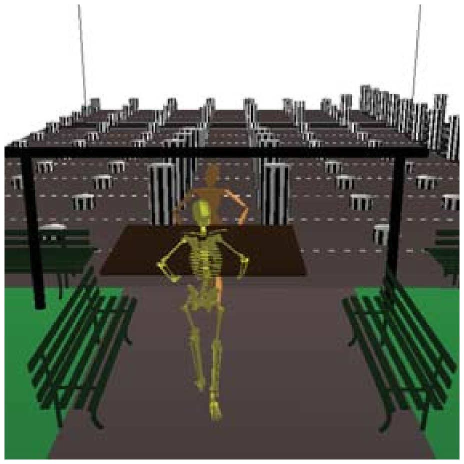

Fig. 15. In our version of Buren's Columns, two virtual mannequins interact to handle a plate.

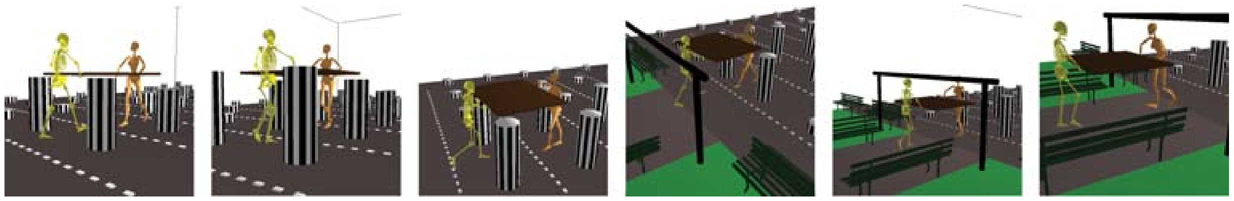

Fig. 16. Selected frames of the resulting animation.

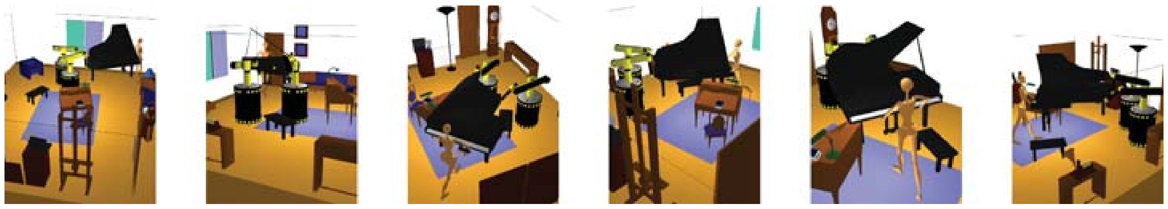

Fig. 17. Two mobile manipulators and a virtual mannequin cooperating to transport a piano.

6.4 Transporting the Piano

In our last example, we treat a version of the piano mover's problem with two mobile manipulators and a virtual mannequin. Here, we want to transport the grand piano in our living room environment. The livingroom is small and contains large objects (tables, a desk, etc.), the walkways are therefore narrow and collisions are likely to occur.

A collision-free path for the 15-DOF reduced system was found between the desk and the piano chair. Figure 17 illustrates some of the synthesized motions for the mobile manipulators and the human mannequin. The virtual robots move away from each other to avoid the stool and then regain their posture.

The cooperation among the characters is ensured during the animation. The 3-dimensionality of the resulting animation is mainly seen in the piano configuration. Figure 18 shows sequenced frames of the planned animation in order to avoid collision between the piano and the desk. In this example, no residual collisions were found.

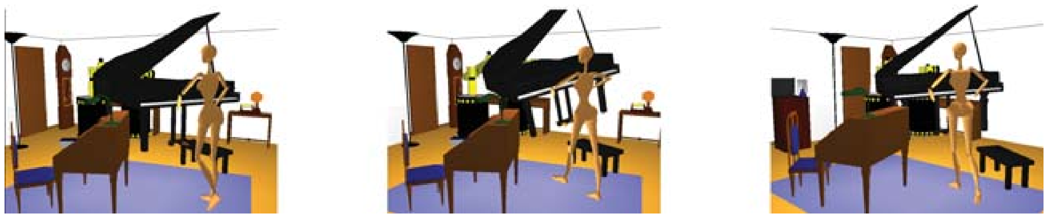

Fig. 18. The piano should be raised in order to avoid collision with the desk.

Table I. Model Complexity (Number of Polygons)

	Environment	System
Office		
- Complete	148,977	17,239
Factory		
- Complete	159,698	18,347
- Col.Test	92,787	
Buren's Columns		
- Complete	44,392	24,837
Living Room		
- Complete	19,077	16,210

Table II. Office (time in seconds)

	In the Office
No. Frames	308 609
Stages	
I. Planner	
- Path	0.5–0.5
- Trajectory	2.0–3.8
II. Animation	
- Locomotion	0.8–1.6
- Manipulation	0.3–0.5
III. Residual Col.	0.2–0.4
Total	3.8–6.80

6.5 Computational Time

The planner has been tested on a workstation Sun-Blade-100 with a 500MHz UltraSparc-IIe processor and 512MB RAM. In Table I, the number of polygons in the different environments as well as the polygons in the system are presented. In the industrial environment, we have identified the polygons that participate in the collision test (i.e., the ones below the mannequin's head level). In the columns and in the living-room environment, all polygons are considered for collision tests purposes.

The required time to compute the examples presented are given in Tables II, III, IV and V (averaged over 100 runs). Here, the results of the planning step are expressed considering a precomputed graph of the environment product of the learning phase. The time taken to build this roadmap was 1.69 seconds, 31.4 seconds, 15.1 seconds and 3.2 seconds for the office, the factory, the columns and the living room, respectively. The time it takes to compute such graphs varies with the complexity of the environment and the fact that it is a probabilistic approach. The tables present only the results of the queries. Two different animations were generated for each trajectory, the second doubling the number of frames.

Table III. Factory (time in seconds)

	In the Factory
No. Frames	268 530
Stages	
I. Planner	
- Path	6.5–6.5
- Trajectory	2.1–4.5
II. Animation	
- Locomotion	0.8–1.6
- Manipulation	0.4–0.8
III. Residual Col.	5.7–11.4
Total	15.5–24.8

Table IV. Columns (time in seconds)

	Buren's Columns
No. Frames	204 405
Stages	
I. Planner	
- Path	0.01–0.01
- Trajectory	2.8–6.1
II. Animation	
- Locomotion	0.6–1.1
- Manipulation	0.6–1.3
III. Residual Col.	3.1–5.6
Total	7.11–14.11

Table V. Living Room (time in seconds)

	Transporting the Piano
No. Frames	78 151
Stages	
I. Planner	
- Path	3.3–3.3
- Trajectory	0.6–1.3
II. Animation	
- Locomotion	0.2–0.5
- Manipulation	0.1–0.3
III. Residual Col.	0.0–0.0
Total	4.2–5.4

Given the significantly higher size and complexity of the industrial environment, it is not surprising that it took more time to compute a collision-free path than the other examples. Note that the time it takes to compute the path at the planning stage is independent of the number of frames in the animation.

In the animation step, the fastest in the algorithm, it is clear that computational time increases proportionally with the number of frames.

The tuning step relies heavily on the complexity of the environment but also on the number of frames where there is a colliding configuration. Note that in the living-room example, the time it takes to tune the animation is zero because residual collisions were not found.

6.6 Videos

Animated sequences of the examples presented in this work can be found at: `http://www.laas.fr/Gepetto/motion-character`.

7. CONCLUSIONS AND FUTURE WORK

We presented an approach to plan and synthesize collision-free motions for virtual mannequins handling a bulky object in a 3-dimensional environment. To accomplish this coordinated task, a geometric and kinematic decoupling of the system is proposed. This decomposition enables us to plan collision-free paths for a reduced system, then to animate the locomotion and grasping behaviors in parallel and finally to clean up the animation from residual collisions. These three steps are applied consecutively by making use of different techniques such as motion planning algorithms, locomotion controllers, inverse kinematics techniques, and path planning for closed-kinematic mechanisms.

At this stage, behavior synthesis is based on the functional decomposition of the system's DOFs (locomotion, grasping, adaptability). The advantage of such decomposition is that the complexity of the system is reduced allowing, for instance, the use of very fast IK algorithms. This decomposition also greatly simplifies the combination of behaviors independently generated. The main limitation of this approach is that the motion planner is not as flexible as it could be. For instance, the characters can not bend to pick the object up from the floor because the grasping DOFs cannot modify the locomotion DOFs. This problem can be solved by using approaches that take into account the whole character's body, such as in Yamane et al. [2004] and Kallman et al. [2003], but at the cost of performance.

As future work, we intend to improve the believability of the resulting animation taking into account the forces working in the environment from the planning stage itself. In this work, we have seen that generated motions are not convincing when the characters handle heavy objects as in the Piano example. Our challenge is thus to combine motion-capture-based control with a physical model of the mannequin. This should be done while preserving as much as possible the current performance of the method.

ACKNOWLEDGMENTS

We thank Juan Cortés for his help on the implementation of the RLG algorithm.

REFERENCES

Badler, N., Erignac, C., and Liu, Y. 2002. Virtual humans for validating maintenance procedures. *Commun. ACM 45*, 7, 56–63.

Badler, N. I., Phillips, C., and Webber, B. 1993. *Simulating Humans: Computer Graphics Animation and Control*. Oxford University Press, Inc., New York, NY.

Baerlocher, P. and Boulic, R. 2004. An inverse kinematics architecture enforcing and arbitrary number of strict priority levels. *Visual Comput. 20*, 402–417.

Blumberg, B. and Galyean, T. 1995. Multi-level direction of autonomous creatures for real-time virtual environments. *Comput. Graph. 29* (Annual Conference Series), 47–54.

Brand, M. and Hertzmann, A. 2000. Style machines. In *Proceedings of SIGGRAPH*. ACM Press/Addison-Wesley Publishing Co., 183–192.

Choi, M., Lee, J., and Shin, S. 2003. Planning biped locomotion using motion capture data and probabilistic roadmaps. *ACM Trans. Graph. 22*, 2, 182–203.

Cortés, J. and Siméon, T. 2003. Probabilistic motion planning for parallel mechanisms. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Earnshaw, R., Magnenat-Thalmann, N., Terzopoulos, D., and Thalmann, D. 1998. Guest editors' introduction: Computer animation for virtual humans. *IEEE Comput. Graph. App. 18*, 5, 20–23.

FALOUTSOS, P., VAN DE PANNE, M., AND TERZOPOULOS, D. 2001. Composable controllers for physics-based character animation. In *Proceedings of SIGGRAPH*. ACM Press, 251–260.

HAN, L. AND AMATO, N. 2000. A kinematics-based probabilistic roadmap method for closed chain systems. In *Proceedings of the International Workshop on Algorithmic Foundations of Robotics (WAFR)*.

KALLMAN, M., AUBEL, A., ABACI, T., AND THALMANN, D. 2003. Planning collision-free reaching motions for interactive object manipulation and grasping. In *Proceedings of ACM SIGGRAPH/Eurographics*, P. Brunet and D. Fellner, Eds. Vol. 22. Eurographics Association.

KALLMANN, M., BARGMANN, R., AND MATARIC, M. 2004. Planning the sequencing of movement primitives. In *Proceedings of the International Conference on Simulation of Adaptive Behavior (SAB)*. Los Angeles, CA.

KAVRAKI, L. E., SVESTKA, P., LATOMBE, J.-C., AND OVERMARS, M. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robotics and Automat. 12*, 4, 566–580.

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

KONDO, K. 1991. Inverse kinematics of a human arm. *J. Robotics Syst. 8*, 2, 115–175.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. In *Proceedings of SIGGRAPH*. ACM Press, New York, NY, 473–482.

KUFFNER, J. 1998. Goal-directed navigation for animated characters using real-time path planning and control. Lecture Notes in Computer Science vol. 1537, 171–186.

LAMIRAUX, F. AND LAUMOND, J.-P. 1997. From paths to trajectories for multi-body mobile robots. In *Proceedings of the 5th International Symposium on Experimental Robotics (ISER)*. Barcelona, Spain, 237–245.

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

LAUMOND, J.-P., Ed. 1998. *Robot Motion Planning and Control*. Springer-Verlag.

LAVALLE, S. M. 1998. Rapidly-exploring random trees: A new tool for path planning. Tech. rep., Computer Science Department. Iowa State University. (Oct).

LAVALLE, S. M., YAKEY, J., AND KAVRAKI, L. E. 1999. A probabilistic roadmap approach for systems with closed kinematic chains. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

LIU, Y. AND BADLER, N. I. 2003. Real-time reach planning for animated characters using hardware acceleration. In *Proceedings of the International Conference on Computer Animation and Social Agents (CASA)*. Washington, DC, IEEE Computer Society, 86.

PARENT, R. 2001. *Computer Animation: Algorithms and Techniques*. Morgan-Kaufmann Publishers.

PERLIN, K. 1995. Real time responsive animation with personality. *IEEE Trans. Visualiz. Comput. Graph. 1*, 1, 5–15.

PETTRÉ, J. AND LAUMOND, J.-P. 2005. A motion capture-based control-space approach for walking mannequins. *Comput. Animat. Virtual Worlds* 16, 1–18.

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

REEDS, J. AND SHEPP, R. 1990. Optimal paths for a car that goes both forward and backwards. *Pacific J. Mathemat. 145*, 2, 367–393.

RENAUD, M. 2000. A simplified inverse kinematic model calculation method for all 6r type manipulators. In *Proceedings of the International Conference in Mechanical Design and Production*. 15–25.

ROSE, C., COHEN, M., AND BODENHEIMER, B. 1998. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Comput. Graph. Appli. 18*, 5, 32–41.

SHILLER, Z., YAMANE, K., AND NAKAMURA, Y. 2001. Planning motion patterns of human figures using a multi-layered grid and the dynamics filter. In *Proceedins of the IEEE International Conference on Robotics and Automation (ICRA)*.

SIMÉON, T., LAUMOND, J.-P., AND LAMIRAUX, F. 2001. Move3d: A generic platform for motion planning. In *Proceedings of the 4th International Symposium on Assembly and Task Planning (ISATP)*.

SIMÉON, T., LAUMOND, J.-P., AND NISSOUX, C. 2000. Visibility based probabilistic roadmaps for motion planning. *Advanced Robotics 14*, 6.

TOLANI, D., GOSWAMI, A., AND BADLER, N. I. 2000. Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical Models 62*, 353–388.

UNUMA, M., ANJYO, K., AND TAKEUCHI, R. 1995. Fourier principles for emotion-based human figure animation. In *Proceedings of SIGGRAPH*. ACM Press, 91–96.

WITKIN, A. AND POPOVIC, Z. 1995. Motion warping. In *Proceedings of SIGGRAPH'95*. ACM Press, 105–108.

Yamane, K., Kuffner, J., and Hodgins, J. K. 2004. Synthesizing animations of human manipulation tasks. In *Proceedings of SIGGRAPH*. ACM Press, New York, NY.

Yamane, K. and Nakamura, Y. 2003. Natural motion animation through constraining and deconstraining at will. *IEEE Trans. Visualiz. Comput. Graph. 9*, 3, 352–360.

Zhao, J. and Badler, N. I. 1994. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Trans. Graph. 14*, 4.

Received October 2004; revised September, October 2005; accepted November 2005