Vis Comput (2012) 28:413–424
DOI 10.1007/s00371-011-0648-x

ORIGINAL ARTICLE

Animating rising up from various lying postures and environments

Wen-Chieh Lin · Yi-Jheng Huang

Published online: 22 September 2011

Abstract Generating rising up motions is an important problem but has less been addressed in computer animation. This problem is challenging as rising motions involve complex motor skills and exhibit wide varieties due to various lying postures and environments. In this paper, we present an approach that utilizes motion planning and dynamics filtering to produce physically plausible rising motions. Our motion planning algorithm connects a given posture to a closest posture in a database of 14 rising motions. Then the dynamics filtering generates a physically plausible motion from a planned motion path. Our experiments show that a variety of motions of rising from various lying postures and different environments with obstacles can be generated easily by our approach.

Keywords Rising motion · Motion planning · Character animation

1 Introduction

Rising up is a very common and important motion for humans. Although there have been many great approaches proposed to animate a wide variety of human motions such as walking, dancing, running, jumping, and manipulating, generating rising motions from various lying postures has rarely been addressed in computer animation. Existing work on rising up animation addressed the problem of reproducing the motion by robustly tracking a reference motion while adapting to different characters and environments [4, 18]. Nevertheless, rising motion exhibits large varieties as it involves complex motor skills and coordination strategy as well as rich interactions with environments [8, 27, 28]. It is important to reflect physical plausibility as well as motion varieties when animating rising motions from arbitrary lying postures. As a first step toward this goal, we address the problem of generating rich rising motions from various lying postures in this paper.

From the view of enriching the style and variety of a type of motions, data-driven approach [11, 16] is very appealing for generating rising motions; however, it is not practical to collect all cases of rising motions considering that a character may rise up from various postures. A good strategy is then combing a motion database that contains some typical rising motions and a motion synthesis approach that can produce the motion variations due to different initial lying postures. According to several biomechanics studies on categorizing the movement patterns of rising motions [27, 28], the varieties of movement patterns mostly present in the phases from lying to sitting or squatting. Thus, it is reasonable and effective to generate a rising motion by synthesizing a transition motion from an arbitrary lying posture to a posture in the rising motion database.

A popular choice for motion synthesis is spacetime-optimization-based approaches [17, 23, 29], which perform very well on many kinds of motions whose dynamic characteristics can be quantitatively described by an objective function and constraints; however, it is difficult to mathematically describe the motor skills and strategy used in rising motion from arbitrary lying postures. Thus, we turn to the motion planning approach for motion synthesis. Motion planning has been used widely in robotics and computer animation for generating locomotion and manipulat-

W.-C. Lin (✉) · Y.-J. Huang
Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan
e-mail: wclin@cs.nctu.edu.tw

Y.-J. Huang
e-mail: sayhello.cs96g@g2.nctu.edu.tw

ing tasks [1, 12, 30]. Nevertheless, rising motion involves lots of contacts with environments and self-collisions between body segments. This makes planning rising motions from arbitrary lying postures more difficult since the configuration space is constrained by obstacles, body segments (self-collision), and joint limits and the feasible path usually needs to pass a narrow passage. Hence, a more sophisticated motion planning approach is needed.

In this paper, we propose an approach to generate motions of rising up from various lying postures by combing motion planning and dynamics filtering techniques. In the motion planning stage, we modified *Rapidly-exploring Random Tree Blossom* (RRT-blossom) algorithm [9] to synthesize a motion path that connects a given initial posture to a similar posture in our rising motion database. A user can also specify an initial posture and a key posture. Our motion planning algorithm can then generate a rising motion that passes these two postures and connects to the motion database. In addition, we develop a loose-to-strict and spatiotemporally local refinement strategy to improve the path-finding capability of rising motion planning. Our motion planning algorithm can plan a rising motion path that avoids self-collisions and static obstacles in an environment. To convert a motion path, which is a sequence of postures, into a physically plausible motion trajectory, we set a smoothed motion path as a reference trajectory for tracking in dynamics simulation. The output of dynamics simulation is the final motion. This process can be conceptually considered as dynamically filtering [31].

The combination of motion planning and motion database in the proposed approach effectively increases the flexibility, richness, and naturalness of the motion, while the dynamics filter further guarantees the physical plausibility of the generated motion. We demonstrate the effectiveness of our approach by testing it on randomly generated lying postures and user specified key postures. Furthermore, we show that our approach can be applied to generating rising motion in environments that are different from those in the motion database. Our user study shows that our results achieve comparable naturalness as human motions. In addition to the applications in character animation, our approach can also be applied to plan/generate reference rising motions for humanoid robots. Through our simulation system, our approach can produce more natural and physically plausible reference motions for humanoid robots.

The contributions of this paper are proposing: (1) a framework that integrates motion planning and dynamics filtering to generate rising motions from various lying postures with rich variations; (2) the loose-to-strict spatiotemporal local refinement strategy that effectively improves path-finding capability of motion planning.

2 Related work

2.1 Computer animation

Generating rising motions has less been addressed in computer animation. Faloutsos et al. [4] illustrate an application of their approach on rising from a supine position. They focused on combining the controllers of different kinds of motion, not generating different rising motions from various postures. Liu et al. [18] recently proposed a sampling-based approach to reconstruct the control underlying a given reference motion. Their approach demonstrates excellent robustness while preserving physical correctness on contact-rich motions including rolling, get-up, and kip-up. This sampling-based approach can produce small motion variations that can be treated as noise; however, reference motions are still needed for producing larger motion variations. Our work can be considered as a complementary module to their approach since rising motion trajectories with rich variations can be easily generated by our approach.

Our motion planner is an RRT-based approach, which has been applied to manipulation planning in computer animation by Yamane et al. [30]. They used motion planning to compute a path of an object to be manipulated. For each planned object orientation and position, the pose of the character is computed to satisfy geometric, kinematic and posture constraints. Instead of planning in the object space, we plan in the posture space using the RRT-blossom algorithm. The concept of connecting a physically simulated motion to a MOCAP motion in our approach is similar to that in Zordan et al. approach [32]; however, they focused on generating dynamic response of MOCAP motions by tracking a desired trajectory, which is formed by linearly interpolating the intermediate postures from the two motion capture sequences before and after the transition. Their approach cannot be applied to our problem since we need to synthesize a trajectory from two postures with large differences. In particular, an arbitrary lying posture or key posture can be very different from any postures in a limited motion database. More importantly, given two postures, our motion planning approach can generate various trajectories, but the liner interpolation approach used in [32] always produces the same trajectory.

2.2 Robotics

Generating rising motions has become an important problem in robotics owing to the rapid development on humanoid robots. Morimoto and Doya [6, 21] proposed a hierarchical reinforcement learning method to generate standing-up movements on a simplified character. Hirukawa et al. [7] and Fujiewara et al. [6] divided a rising motion into several contact states and used a contact-state graph to represent them.

This approach works well on robots, but it is difficult to define a proper contact-state graph for human motions rising from various lying postures. Kanehiro et al. [10] generated getting-up motions by linearly interpolating any given lying posture to its most similar posture in a predefined falling state graph. Their work focuses on generating a smooth sequence of rising postures while we address on producing a physically plausible rising motion. Besides, our approach can generate motions with more varieties and flexibilities since we use a motion planning algorithm instead of linear interpolation and allow a user to specify a key posture.

2.3 Biomechanics

McCoy and VanSant [20] and Ford-Smith and VanSant [5] compared movement patterns of people rising from a bed in different ages. For adolescents, they developed four categories of movement patterns: far upper extremity, near upper extremity, axial region and lower extremities. In the age between 30 and 59, they developed four categories of movement patterns: left upper limb movement patterns, right upper limb movement patterns, head and trunk movement patterns and lower limb movement patterns. They experimented and computed the probability of each movement pattern. The goal of these biomechanics studies is to analyze rather than generate the rising motion.

3 Background on motion planning

Motion planning is used to search the system configuration space of one or more geometric bodies for a collision-free path that connects a given start and goal configuration while fulfilling constraints imposed by obstacles. We adopt and modify the rapidly-exploring random trees blossom (RRT-blossom) algorithm [9] to solve our motion planning problem due to its computational efficiency and path-finding capability.

3.1 Rapidly-exploring random tree (RRT)

The rapidly-exploring random tree (RRT) [14] consists of an efficient data structure and sampling scheme that can quickly search high-dimensional configuration spaces, which may have both algebraic constraints arising from obstacles and differential constraints arising from nonholonomy and dynamics. The key idea of the RRT is to bias the exploration toward unexplored portions of the configuration space. At each iteration, the algorithm attempts to extend the RRT by adding a new node that is biased by a randomly-selected configuration x_{rand}. The right subtree in Fig. 1 illustrates this EXTEND operation, which selects the nearest node x_{near} in the existing RRT to a given sample configuration x_{rand} and proposes a new node x_{new} by moving toward

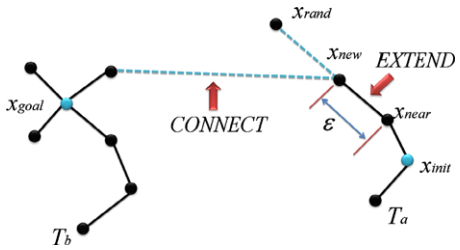


Fig. 1 EXTEND operation in RRT-connect

x_{rand} with some fixed incremental distance ε. x_{new} is tested for collision. Three situations may occur during this movement: *Reached*, x_{rand} is directly added to the T since x_{new} is within ε of x_{rand}; *Advanced*, where a new node x_{new} is added to the T; *Trapped*, where the proposed new node is rejected because it does not lie in a collision-free space.

Although RRT performs well and is simple to implement, it suffers from slow convergence rate. To improve RRT's computational efficiency, Kuffner and LaValle developed the RRT-connect algorithm [13] for the path planning problem involving no differential constraints. Their approach is based on two ideas: the CONNECT heuristic that attempts to move over a longer distance, and the growth of two RRTs from both initial node x_{init} and goal node x_{goal} by alternatively executing the EXTEND and CONNECT operations. Figure 1 shows the basic operation of the RRT-connect algorithm. Instead of extending an RRT by a single step, the CONNECT heuristic greedily explores the space by repeating the EXTEND step until x_{goal} or an obstacle is reached. The CONNECT operation thus allows a much larger movement than the EXTEND operation.

3.2 RRT-blossom

RRT-blossom [9] is a variant of the RRT-connect. It performs well on both loosely-constrained and highly-constrained environments. The key idea of RRT-blossom is an implicit flood-fill-like mechanism, which is particularly suited for escaping from local minima in highly constrained problems. The main difference between RRT-connect and RRT-blossom is the EXTEND function. Figure 2 shows the EXTEND function used in RRT-blossom, where the SIM function instantiates a new node by moving a sampled node x_{near} with a control input u. u is selected from U, which defines all possible control actions. FAILURE function checks if the transition from x to x_{new} incurs a collision or violates any global constraints. The implementation of SIM and FAILURE is application-dependent. We discuss how we implement these two functions in Sect. 4.2.

RRT-blossom can explore a configuration space more quickly because it adds multiple new nodes by spanning a subtree within a range rather than a single node at each time. To avoid searching all spanning nodes in the subtree, RRT-blossom utilizes a regression operation to regress the surplus

```
function BUILD_RRT_BLOSSOM(x_init, x_goal)
  T_a.init(x_init)
  T_b.init(x_goal)
  while TIME_ELAPSED() < MAX_TIME do
    x_rand ← RANDOM_CONFIG()
    x_a ← EXTEND_BLOSSOM(T_a, x_rand)
    if x_a then
      x_b ← EXTEND_BLOSSOM(T_b, x_a)
      if x_b then
        if dist(x_a, x_b) < ε then
        return PATH(T_a, T_b)
    SWAP(T_a, T_b)
  return Failure

function EXTEND_BLOSSOM(T, x)
  New_node_added = False
  x_near ← NEAREST_NEIGHBOR(T, x)
  for u ∈ U do
    x_new ← SIM(x_near, u)
    if FAILURE(x_near, x_new, u) then
      next u
    if REGRESSION(T, x_near, x_new) then
      next u
    T.add_node(x_new)
    T.add_edge(x_near, x_new)
    New_node_added = True
  if New_node_added then
    return the new node closest to x
  return False

function REGRESSION(T, x_parent, x_new)
  for node n ∈ T do
    if dist(n, x_new) < dist(x_parent, x_new) then
      return True
  return False
```

Fig. 2 RRT-blossom algorithm

nodes that are close to other nodes in the existing tree. Figure 3 illustrates the blossom and regression operations in the RRT-blossom algorithm. The regression operation improves the performance of RRT-blossom algorithm; however, it induces a new problem that all viable paths may be regressed if the blocked expansion incidentally lies close to a narrow passage as shown in Fig. 4 of [9]. RRT-blossom solves this *viability* problem by labeling an edge as one of the following four states: *untried*, *live*, *dormant,* or *dead*. Figure 4 shows the progression of the viability states of an edge. The *untried* edges are ones that have not yet been considered for expansion. They become *live* upon instantiation. An edge is marked *dormant* if the expansion is not allowed due to regression. Finally, edges that have been found to have collision in space are marked *dead*. RRT-blossom only searches

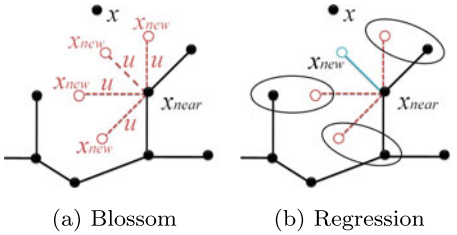


Fig. 3 (**a**) The possible expansions (*red dashed edges*) for a node x_{near}. (**b**) All the *red expansions* are regressing since a foreign node is closer than the parent (these occurrences are indicated with *ellipses*). Only the *blue edge* is suitable for instantiation in this case

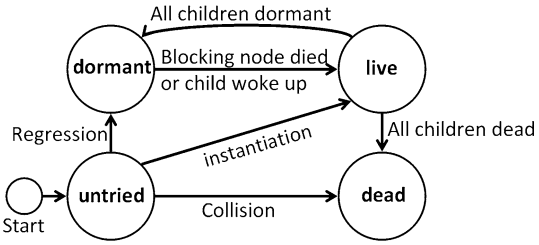


Fig. 4 Progression of the viability state of an edge

the edges in *live* state. There are two ways that a *live* edge may change their state. One is all children dormant, and the other is all children dead. When an edge close to obstacles is dead, the edge's parent may be dead, and the path towards the obstacles also disappears. This edge-receding mechanism prevents RRT-blossom from being trapped in local minima.

4 Approach

Figure 5 shows an overview of our approach, which consists of three main stages: *connecting posture selection*, *motion planning* and *dynamics filtering*. Given an initial posture P_{init} specified by a user, the most similar posture to P_{init} in the rising motion database is automatically found in the connecting posture selection stage. The selected posture is denoted as P_{con}. If the user wants to have more control on the generated rising motion, our approach also allows the user to specify P_{con} as a key posture. In the motion planning phase, we plan a motion to connect P_{init} and P_{con}. Finally, in the dynamics filtering phase, we further refine the planned motion by smoothing and dynamics filtering to ensure the physical plausibility.

4.1 Connecting posture selection

A posture is defined by $P = \{p, q, V\}$, where p is the global position of the root, q is the global orientation of the root represented in quaternion, and $V = (v^1, \ldots, v^n)$ is a vector that describes the joint angles of all joints of the skeleton

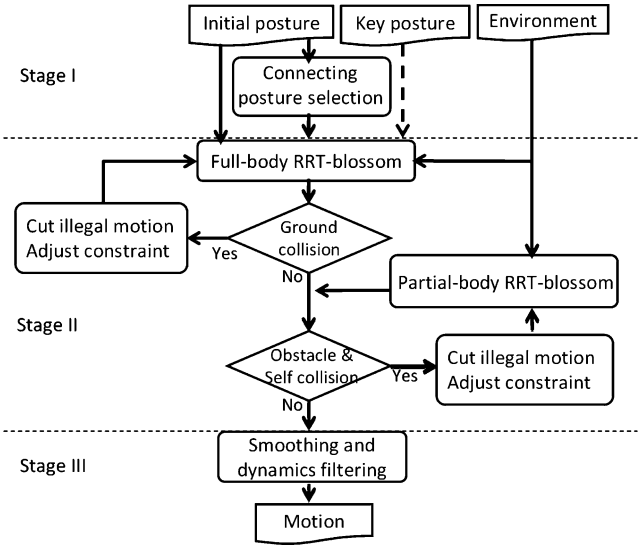


Fig. 5 Our approach consists of three stages: posture selection, motion planning, and dynamics filtering. Note that the input of key posture is optional. If a key posture is specified by a user, it is used as a connecting posture in the motion planning phase

excluding the root joint. i denotes the ith DOF of a skeleton and v^i is represented in an Euler angle. One may consider using quaternion or exponential map to represent all 3D joints; however, quaternion would increase the dimension of V causing the difficulty of motion planning while the exponential map takes more computational time in motion planning since it needs special treatment for the blossom and extend operations. Therefore, to avoid the gimbal lock problem that may occur at the root joint more likely and make the implementation of RRT-blossom easier, we use quaternion to represent the rotation at the root joint and Euler angle to represent the other joints.

To measure the difference between two postures $P_1 = \{p_1, q_1, V_1\}$ and $P_2 = \{p_2, q_2, V_2\}$, we define a posture distance metric as follows:

$$\mathrm{dist}(P_1, P_2) = w_q \cdot \mathrm{dis}\, Q(q_1, q_2) + \mathrm{dis}\, V(V_1, V_2), \tag{1}$$

where the first term measures the difference between the global orientation of the root joints of P_1 and P_2; the second term computes the posture differences at the other joints. w_q weights these two terms, which are defined as

$$\mathrm{dis}\, Q(q_1, q_2) = \left\| \log\left({q_1}^{-1} q_2\right) \right\| \tag{2}$$

$$\mathrm{dis}\, V(V_1, V_2) = \sqrt{\sum_{i=1}^{n} w^i \cdot \left(v_1^i - v_2^i\right)^2}. \tag{3}$$

w^i's are the weighting of the importance of each joint. v_1^i and v_2^i are the Euler angle of the ith DOF of P_1 and P_2, respectively. Note that the global position and facing direction of P_1 and P_2 are roughly aligned before their posture difference is computed.

The posture difference metric defined in (1) is used to find the connecting posture in our motion database. To increase the search speed, we apply the K-means algorithm to cluster all postures in our motion database into N groups. We then use an N-ary tree to represent these groups. Each node of the tree stores a group of postures and the root node contains the representative posture of each group. The representative posture of a group is the mean of all postures in the group. Once the tree of clustered postures is constructed, our posture selection process can be efficiently performed.

4.2 Motion planning

Motion planning is applied to generate a motion path that connects an initial posture P_{init} to a connecting posture P_{con}. Our motion planning approach is developed based on the RRT-blossom algorithm; however, as RRT-blossom is originally applied to a 2D path planning problem, we make several modifications to RRT-blossom so that it can be applied to high-dimensional motion planning for human characters. At the low-level, we modify the EXTEND function in RRT-blossom to handle posture data consisting of different physical quantities and take into account joint limit constraint, obstacle avoidance and self-collision check under the framework of RRT-blossom. At the high-level, we propose a loose-to-strict and spatiotemporally local refinement planning strategy to improve the path-finding capability of motion planning.

4.2.1 *Modifications of RRT-blossom to handle posture space*

We modified the new node generation procedure in the SIM function of RRT-blossom (Fig. 2). In RRT-blossom, u is set as a constant moving step ε whose direction is randomly chosen within a spanning angle of the main direction from x_{near} to an intermediate goal x as shown in Fig. 3(a). In our implementation of RRT-blossom, a node representing a posture is defined as $x = \{p, q, V\}$, which consists of different types of physical quantities. We need to plan all 6 DOFs of the root joint if P_{init} and P_{con} differ in the root joint's 6 DOFs. This usually happens when P_{con} is a user-specified key posture. We first plan a path for the global position of the root using the conventional RRT-blossom. The remaining global orientation and joint angles are then planned together using the modified RRT-blossom algorithm. We expand the subtree of different physical quantity separately in the node blossom operation. We first compute a new node $x_{\mathrm{new}} = \{p_{\mathrm{new}}, q_{\mathrm{new}}, V_{\mathrm{new}}\}$ toward a sample node $x = \{p, q, V\}$ using (4) and (5),

$$q_{\mathrm{new}} = \mathrm{slerp}(q_{\mathrm{near}}, q, t), \quad t = \frac{\varepsilon_q}{\mathrm{dis}\, Q(q_{\mathrm{near}}, q)}, \tag{4}$$

$$V_{\mathrm{new}} = V_{\mathrm{near}} + \frac{\varepsilon_e}{\mathrm{dis}\, V(V_{\mathrm{near}}, V)} \cdot (V - V_{\mathrm{near}}), \tag{5}$$

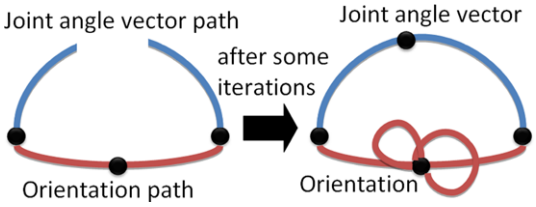


Fig. 6 Illustration of the unsynchronized termination problem. (*Left*) When the global orientation path connects, the joint angle vector path has not connected yet. (*Right*) When the joint angle vector path connects, the global orientation path has generated some surplus segments

where x is an intermediate goal randomly generated; ε_q and ε_e are the fixed moving step of u for global orientation q and joint angle vector V, respectively. Other new nodes are then instantiated (or blossom) within a spanning angle of the first new node. For a joint angle vector, a span of ± 15 degrees for each DOF toward the intermediate goal direction is used in our approach.

Planning a full-body motion consisting of global orientation and joint angles can cause an unsynchronized termination problem in which the path for global orientation and joint angles may connect the subtrees from two ends at different iterations as illustrated in Fig. 6. To resolve this problem, the connected goal value will be fixed as long as either the global orientation or joint angle vector reaches the final goal value while only the remaining unconnected part is planned.

4.2.2 Joint limit constraint and collision detection

We impose the joint limit constraint in the SIM function, i.e., any new node generated by the SIM function should satisfy the joint limit constraint. Additionally, we check if any body segment collides with ground, obstacles or other body segments in the motion planning process. This is implemented in the FAILURE function in Fig. 2. For computational efficiency, Sphere-Swept Volume (SSV) is applied for collision detection [3].

4.2.3 Loose-to-strict spatiotemporally local refinement strategy

Our motion planning algorithm consists of full-body and partial-body motion planning. The former plans the full body motion that avoids ground penetration, while the latter focuses on planning the motion of those body segments that collide with obstacles or other body segments (self-collision). An iterative refinement strategy is adopted in both full-body and partial-body motion planning. We repeat the RRT-blossom algorithm to refine the planned motion by gradually reducing the tolerance of violation of constraints until all constraints are satisfied.

At the first iteration, we set the horizon of the ground at the lowest position of all body segments of P_{init} and P_{con} and reduce all obstacles' size to zero by setting the radius of SSV of each obstacle as zero. That is, there are no obstacles and ground in the environment initially and the motion planner can generate a collision-free motion easily. We then gradually enlarge the obstacle size or raise the horizon of the ground at each iteration of motion planning until the obstacle size and ground level meet the originally set values and there is no collision in the generated motion. In our experience, setting the incremental of SSV radius and ground level per iteration at 0.025 works well. Thus, 0.025 is used in all of our experiments. Loose-to-strict spatiotemporally local refinement strategy makes our motion planning more flexible and efficient as it can be performed locally both in space (partial-body motion planning) and time (illegal motion segment replanning).

4.2.4 Cut and merge illegal motion segments

In full-body and partial-body motion planning, if any part of the character collides with the ground or any obstacle, the posture of the character is considered as an illegal posture and those motion frames containing illegal postures are called an illegal motion segment. We cut an illegal motion segment from a planned motion path and then insert a new motion segment replanned by the RRT-blossom algorithm. If there are multiple illegal motion segments in a planned motion, we merge and replan those illegal motion segments that are temporally close. Our cut-and-merge process reduced the complexity of motion planning and improves the smoothness of the planned motion.

It is noteworthy that RRT-blossom finds a collision-free path instead of the shortest path. This increases the variability of planned motions since different posture sequences can be generated given the same P_{init} and P_{con}. On the other hand, the regression operation avoids similar postures to be added to the planned path. This implies that there is no "going back" in the path and might slightly reduce the variability of generated motions, e.g., high-frequency motions like quiver may be forbidden. Nevertheless, as high frequency noise does not appear often in natural motions, this restriction does not harm the naturalness of planned motions.

4.3 Smoothing and dynamics filtering

A result planned by RRT-blossom is simply a sequence of postures. A path formed by directly connecting these postures is not smooth and does not contain temporal information. Therefore, we need to generate a physically plausible motion trajectory from this posture sequence. We use cubic Bézier curve to generate a smoothed path by treating the planned posture sequence as control points. All postures are represented in quaternion when constructing this

Bézier curve. We then generate a motion trajectory from the smoothed curve using a velocity profile of constant speed. This trajectory is a rising motion that is smooth but not physically correct.

We treat the smoothed motion trajectory as a reference motion trajectory and apply tracking control and forward dynamics simulation to generate a physically-plausible motion. We call this process *dynamics filtering* since it is conceptually similar to the dynamics filter proposed by Yamane and Nakamura [31]. We adopt Open Dynamics Engine (ODE) for forward dynamics simulation and use the velocity-driven control method [25] to track the reference motion. Additionally, we apply the virtual actuator control [22] to assist trajectory tracking. The basic idea is to apply an "external" force generated by a virtual actuator to control a body segment. This external force is then converted into internal joint torques distributively generated by those joints that would affect the motion of the body segment to be controlled. In our implementation, we control the center of mass of the body by converting a control force into joint torques of the joints of limbs that have contact with the environment. This is helpful for maintaining balance while rising up. Using velocity-driven control and virtual actuator control, we are able to track all planned and smoothed motion trajectories from lying to sitting or squatting successfully and then connect to a closest squatting-to-standing trajectory in our MOCAP database. To increase the smoothness of this connection, we search between a simulated motion and MOCAP motion database to find two postures from each of them that have the smallest difference. We then connect the simulated motion to a MOCAP rising motion from the most similar frame. Motion blending is applied to those frames around the connecting frame. This connecting frame is constrained to be located after the frame of the key posture so that the user-specified key posture can be preserved in the final motion.

Although our dynamics filter mainly works on rising motions from a lying state to a sitting or squatting state, the effect of dynamics filtering is still significant. We demonstrate several examples that reflect the motion variations due to the change of physical properties in our results. In fact, according to the component actions summarized in [27, 28], a rising motion can be roughly divided into four phases: (1) lying and/or rolling, (2) moving to sitting (supine/lateral) or getting up on all fours (prone/lateral), (3) squatting or half kneel, and (4) standing up. The first two phases exhibit larger variations and more varieties [26, 28]. Therefore, it is actually reasonable to focus on the first two phases in terms of generating rising motions with varieties and flexibilities. Besides, similar to the idea promoted in [32], it is computationally more efficient to exploit MOCAP data as much as possible since motion planning and dynamics filtering needs additional computation.

5 Experimental results

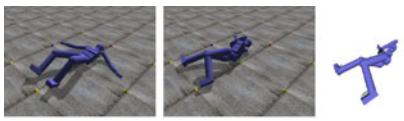
We collect 14 rising motions including rising from 6 supine positions, 4 prone positions, and 4 lateral positions from the CMU MOCAP database to construct our motion database. All these motions are rising from flat ground without obstacles. The weight of each joint for the posture difference (equations (1) and (3)) are set as follows: root = 5, head = 0.5, wrist = 0.5, hand = 0.3, fingers = 0.1, thumb = 0.1, foot = 0.7, toes = 0.1, and all the other joints = 1. The number of groups in the K-means clustering is $N = 50$ in our experiments and the size of groups ranges from 5 to 2442. The dynamics filter is performed by the Open Dynamics Engine (ODE) version 0.11. All the experiments were run on a 2.66 GHz Intel Core2 Duo CPU E6750 computer with 3GB of RAM. The average computational time of motion planning and dynamics filtering with nonoptimized code is about 11.87 seconds and 5.3 seconds, respectively. In the accompanying video, we labeled in each motion the portion which is planned and dynamics filtered.

5.1 Rising up from random initial postures and key postures

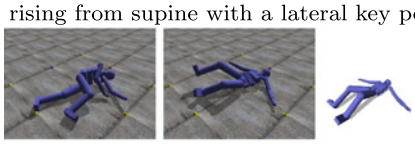
We first test our algorithm in an obstacle-free environment. We randomly generate 60 initial lying postures for testing, include 20 in supine position, 20 in prone positions and 20 in lateral position. Motion planning from all 60 lying postures are successfully done. Figure 7 shows some of these testing postures in our experiment. As our posture selection method often finds a connecting posture close to an initial posture, the planned and dynamics filtered motion usually only appears a short period in the generated animations when only an initial posture is given. In other cases when an additional key posture is given or there are obstacles, the portion of planned and dynamics filtered motion increases considerably.

Our approach also allows a user to specify a key posture in addition to an initial posture. Figure 8 shows some of our results of rising from supine, lateral, and sitting postures. One can find that key postures are closely performed in these examples. More results can be seen in the accompanying video. There are some cases that the key posture is not achieved since the dynamics filtering in our approach prefers physical plausibility to user's control. This is good for novice users as they may assign a key posture violating dynamical constraints, but our approach can still produce a physically plausible motion while matching the key posture as close as possible. Note that if a key posture is not a posture in our motion database, motion planning is applied again to connect the key posture to a posture in our motion database. This provides the user more flexibility to choose a desired key posture. On the other hand, we can also restrict a

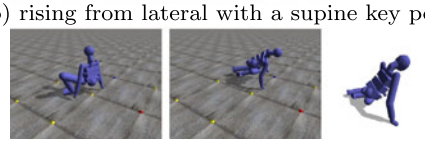
Fig. 7 Some of the random initial postures used in our experiment



(a) rising from supine with a lateral key pose.



(b) rising from lateral with a supine key pose



(c) rising from sitting with a hand-push key pose

Fig. 8 Snapshots at the initial and key frames of three rising motions. The images at the right column shows the key posture specified by the user. The key posture is closely followed in the generated motions

user to pick a key posture from the existing motion database. This may reduce the range of key postures that a user can choose if the size of the motion database is limited; however, it also provides a good guidance for the user to select a more natural posture. Whether a key posture is in the database or not, our approach can generate a smooth rising up motion that connects the initial and key postures. In our opinion, if the database of rising motion is large enough, then picking a key posture from the motion database might be a better choice; however, if the size of the motion database is limited like ours, allowing the user to arbitrarily specify a key posture is more feasible solution.

5.2 Rising up in different environments

Although our motion database only contains rising up on flat ground without obstacles, our approach can generate various rising motions that adapt to different environments. Figure 9 shows an example that a character rises up from a prone position with face on a wood beam. One can see how the character adjusts his hand motion to utilize the wood beam for supporting his upper body while avoiding collision with the wood beam. Figure 10(a, b) shows rising up from a low bed with two different lying positions: one with upper legs outside the bed and the other on the bed. The character's arm motion adapts to the lying position by pushing the bed and ground with different parts of his arms and hands. In this example, only the initial lying posture is given. In Fig. 10(c), we simulated rising up from a soft bed by setting the contact model between the body and the slope as a soft constraint, i.e., a virtual soft spring to penalize penetration. The initial lying posture and position are the same with those at (a). In Fig. 11, we demonstrate rising up from a lateral position under a table. An initial posture and a key posture lying laterally with the head facing outward are given, our approach generates a physically plausible motion without collision with the environment.

In addition to normal ground, our approach can also simulate rising up on different ground conditions. Figure 12 shows a comparison of rising up on icy and normal ground. In the accompanying video, one can observe that the character's body slips on icy ground and the character eventually face a different direction after standing up. Finally, we tested our approach on rising up from lying on a slope. Figure 13 shows several snapshots of our result. This result demonstrates that our approach can also adapt the rising motion to a slope environment. The examples in Figs. 9–13 illustrate that despite different environments, our motion planning approach can still generate a collision-free path and dynamics filtering can refine the path to produce physically plausible motion that reflects the change of the environment.

5.3 Motion retargeting

Our approach can be used for motion retargeting. Figure 14 shows the motion variations at the same time frame due to the change of character's body: (a) original character; (b) shoulder is two times as wide as the original; (c) arms are 1.6 times as long as the original; (d) arms and upper body are 1.6 times and 2.2 times long as the original, respectively. Note that the mass and inertia of the corresponding body segments are also changed according to these bone length changes. One can see the effects of dynamics filtering on the character's motions in the accompanying video.

5.4 User study and validation

We conducted a user study to evaluate the naturalness of our results. Table 1 summarizes our user study result. We prepared 25 animations including 7 motions from MOCAP database and 18 motions from our results. Each subject watched all 25 animations, which were played in a random order on a computer screen, and gave a score to each motion based on the naturalness. The range of score, from best to worst, is 10 to 1. There are 27 males and 13 females aged

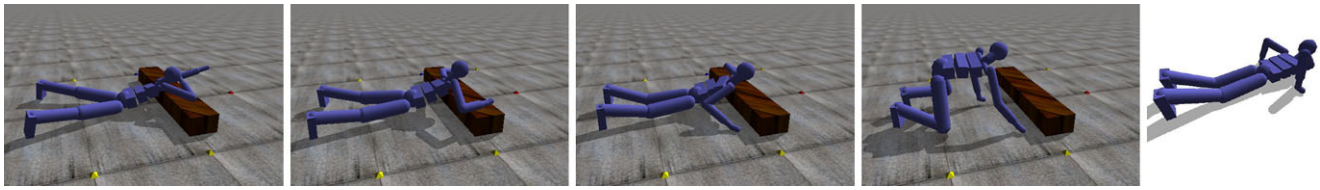


Fig. 9 A rising motion with a wood beam below the face. One can see that the upper arm movements in the generated motion adapt to the wood beam. The *leftmost* and *rightmost images* are the initial and key posture, respectively

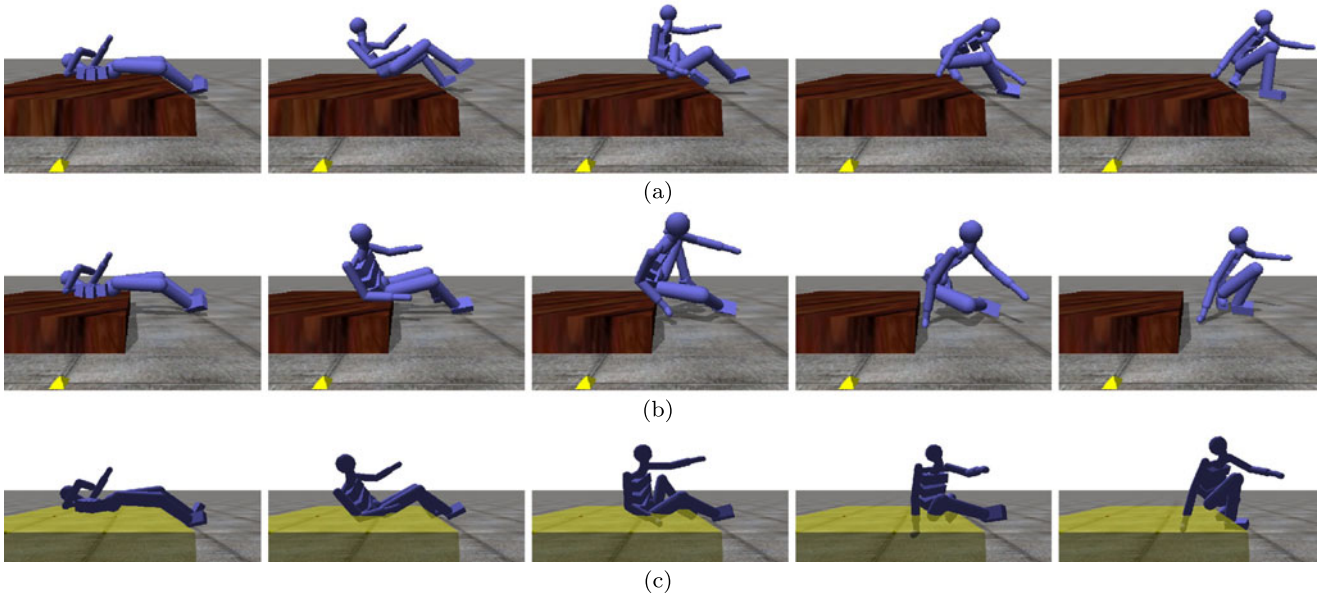


Fig. 10 Rising from a low bed with different lying positions and rising from a soft bed. Only the initial lying posture is given in this example. (**a**) Upper and lower legs are outside the bed. (**b**) Only lower legs are outside the bed. The character's arm motion adapts to the lying position by pushing the bed and ground with different parts of his arms and hands. (**c**) Rising from a soft bed. The initial lying posture is the same as that on the *top row*. One can observe how the rising motion adapts to a soft surface

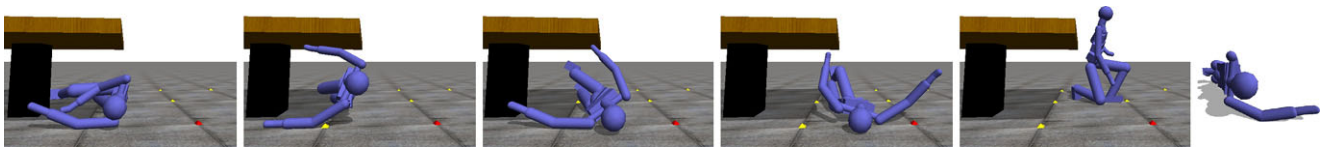


Fig. 11 A rising motion from an initial posture under a table. The *leftmost* and *rightmost images* are the initial and key postures, respectively

between 19 to 60 participated in the study. 21 of them have graphics research background. The average score and standard deviation of MOCAP and our results are (6.92, 2.22) and (6.29, 2.27), respectively. It appears that the naturalness of our results is slightly worse than, but perceptually comparable to that of human motions (MOCAP data). To validate the effectiveness of our loose-to-strict spatiotemporally local refinement strategy, we compare the average computational time of motion planning used in Figs. 9–11. One can find in Table 3 that although our refinement strategy increases the computational time slightly, it improves the path-finding capability effectively.

To further verify the effect of motion planning and dynamics filtering, we conducted another user study as shown in Table 2. We compare our results with those generated by removing dynamics filtering from our approach and removing motion planning from our approach. Specifically, for the approach without dynamics filtering, we only use motion planning and smoothing o produce rising motions, while for the approach without motion planning, we only use linear interpolation and dynamics filtering to generate rising motions. We performed our user study using the method of paired comparisons [2]. In this method, items are presented side-by-side in pairs to a human subject, who then selects a

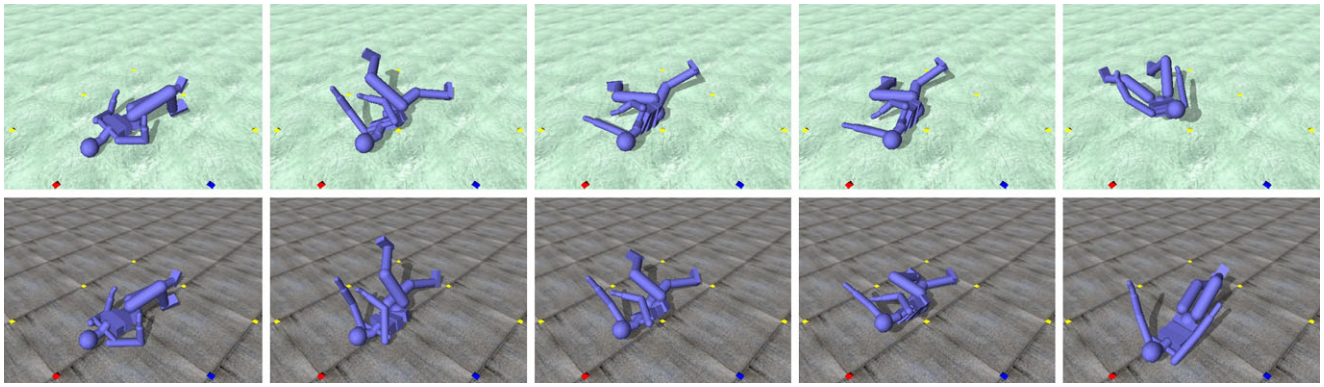


Fig. 12 Comparison of rising up from icy ground (*top*) and normal ground (*bottom*). The character on the *top row* was slipping on icy ground and going to stand up toward a different direction

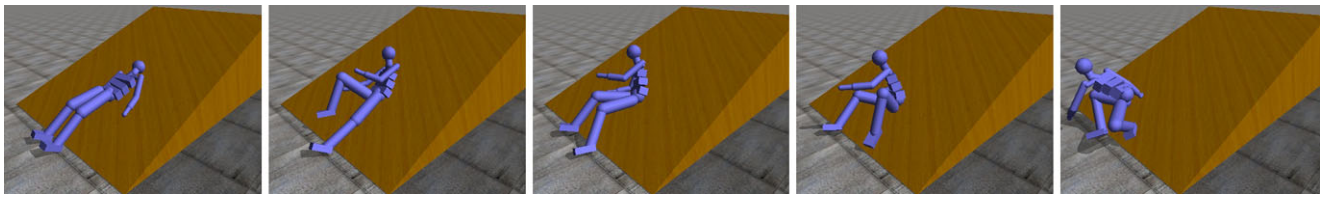


Fig. 13 Rising up from lying on a slope. One can see the character's motion adapts to the slope and presents physical plausibility

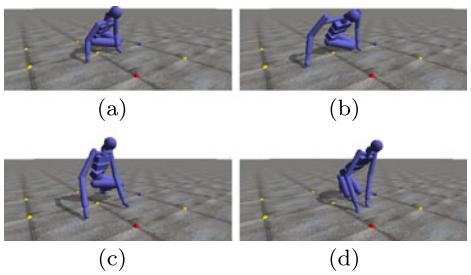


Fig. 14 Example of motion retargeting. A snapshot of each motion at the same frame is presented: (**a**) original character; (**b**) shoulder is two times as wide as the original; (**c**) arms are 1.6 times as long as the original; (**d**) arms and upper body are 1.6 times and 2.2 times long as the original, respectively. One can observe from the accompanying video that the resulting motions are dynamically different

preferred one in each pair. Following this method, we prepared a web-based survey showing pairs of rising motion videos. For each pair, a subject needs to vote which one looks more natural. There were 24 subjects participating in our user study including 17 males and 7 females, 13 (of 24) subjects have graphics background. We compared on rising from 3 different lying postures, so each approach was compared $3 \times 24 = 72$ times. To reduce the bias in our user study, we randomized the playing order of video pairs and provided only the most necessary information to the subjects.

Table 2 shows the overall preference by summing the votes for each method across subjects. It shows that our approach is clearly preferred over that without dynamics filtering since our results were favored in 73.61% (53 of 72) of the comparisons with the approach without dynamics filtering. When compared with the results without motion planning, our results were still preferred in 72.22% (52 of 72). Overall, the subjects favored our method in 48.61% (105 of 216) of the comparisons, while the preference for the results without dynamics filtering and the results without motion planning is are 28.24% (61 of 216) and 23.15% (50 of 216), respectively. The interobserver variability, Kendall's coefficient of agreement, is $u = 0.1375$ for Table 2, with a p-value <0.01. Hence, there is a statistically significant agreement among the subjects regarding the three approaches. We refer readers to [2] for a detailed explanation of these indicators. This user study verifies that both dynamics filtering and motion planning are needed for generating natural rising motions. It also shows motion planning has slightly more influence than dynamics filtering for generating natural rising motions.

5.5 Discussion and limitations

It is an interesting question that whether the closest key posture is the best choice or not. We may verify this issue by leaving one MOCAP motion out and then compare our result with the MOCAP motion; however, this validation would need a larger database, which we do not have currently. To make the transition strategy closer to the natural strategy, we can incorporate some findings in biomechanical studies that analyze the strategies used in rising up motion. In our implementation of motion planning, the

Table 1 User study on the naturalness of rising animations from MOCAP and our approach. The score range is from 10 (best) to 1 (worst). From the 2nd to 6th columns are: MOCAP, rising up from an initial posture, from an initial and a key posture, with obstacles, from a sitting posture. The 3rd row lists the total number of each type of motion

Source	MOCAP	Our results			
Type	No-obstacle	No-key	Key	Obstacle	Sitting
Num	7	2	9	4	3
Mean	6.92	7.06	6.23	6.39	5.82
Std	2.22	2.50	2.20	2.23	2.26
Total	6.92, 2.22	Mean = 6.29, Std = 2.27			

Table 2 Preferences of 24 subjects for 3 motion generation approaches, e.g., an entry n in row 1 and column 2 means the result of approach 1 was preferred n-times to the result of approach 2

	1	2	3	Total (216)
1. Our approach	–	53	52	105
2. Without dynamics filtering	19	–	42	61
3. Without motion planning	20	30	–	50

computational time of each iteration of loose-to-strict refinement ranges from 0.41 to 4.9 seconds and its average is about 0.787 seconds. In our experiments, we first try to plan a motion satisfying the original (strict) constraints. If a collision-free path cannot be found, we then apply the loose-to-strict refinement strategy. Currently, we used a fixed incremental for simplicity, so the total number of iterations is 40. Dynamically adjusting the incremental at each iteration is a better strategy that can improve the computational efficiency.

It may also be possible to first plan multiple paths from P_{init} to P_{con} and then select the best one according to motion naturalness; however, it is still an open question of measuring the naturalness of a motion. Choosing the best planned path based on the physics compatibility is feasible, but our dynamics filtering at the later stage would actually ensure the physics compatibility. In fact, our motion planning currently only consider lower-level constraints, such as obstacle avoidance and joint limit. Higher-level constraint, such as naturalness or coordination strategy may be included in the motion planner.

Currently, our dynamics filtering mainly works on rising motions from a lying state to a sitting or squatting state. To capture more dynamic variations of rising motions, especially balancing behaviors from sitting/squatting to standing, it is desirable to simulate the whole motion from lying to stance states. Several recent approaches [18, 19, 24] can be applied to improve our dynamics filtering. We will explore the above issues about motion planning and dynamics filtering in the future.

Table 3 Comparison of average computational time (in seconds) of motion planning without and with loose-to-strict spatiotemporally local refinement

Figure	9	10 (top)	10 (bottom)	11
Without	Fail	2.85 s	2.63 s	Fail
With	6.18 s	2.92 s	3.06 s	20.45 s

6 Conclusion and future work

We present a simple and effective approach to generate rising motion from various lying postures using motion planning and dynamics filtering. In the motion planning stage, we use a modified RRT-blossom algorithm and design a loose-to-strict spatiotemporally local refinement strategy to plan a path connecting a given posture and motion database. In smoothing and dynamics filtering stage, velocity-driven control and virtual actuator control are applied in a dynamics simulator to track a motion trajectory constructed from the planned motion path. Our experiments show that a variety of motions of rising from various lying postures and different environments with obstacles can be generated by our approach easily. In particular, our motion planner can generate a collision-free path that adapts to the environment while dynamics filtering can produce physically plausible motions with variations that reflect the change of physical properties.

In the future, we would like to add contact-dependent constraints that only some DOFs are allowed to move. For example, if a character lying prone with his arm under his body, the arm cannot move unless it pushes the ground. Besides, the samples in our rising motion database may not reflect the real distribution of human motions found in [27, 28]. If more rising motions collected according to the statistic of human rising motions, the naturalness of our results can be further improved. Also, we currently use a constant speed profile to generate a motion trajectory from a planned posture sequence. A better way to specify the speed profile may be based on the transition postures and the selected MOCAP. Furthermore, we may consider to apply kinodynamic planning [15] to enforce dynamical constraints in the motion planning stage if the curse of dimensionality problem can be solved. Finally, we would like to improve the computational performance of our approach so that it can be applied to real-time applications such as games or virtual training.

Acknowledgements This work was supported in part by National Science Council under Grant NSC-99-2628-E-009-178- and the UST-UCSD International Center of Excellence in Advanced Bioengineering sponsored by the Taiwan National Science Council I-RiCE Program under Grant No. NSC-99-2911-I-009-101.

References

1. Choi, M.G., Lee, J., Shin, S.Y.: Planning biped locomotion using motion capture data and probabilistic roadmaps. ACM Trans. Graph. **22**(2), 182–203 (2003)
2. David, H.A.: The Method of Paired Comparisons. Charles Griffin & Company Ltd, London (1988)
3. Ericson, C.: Real-Time Collision Detection. Morgan Kaufmann, San Mateo (2005)
4. Faloutsos, P., van de Panne, M., Terzopoulos, D.: Composable controllers for physics-based character animation. In: SIGGRAPH, pp. 251–260 (2001)
5. Ford-Smith, C.D., VanSant, A.F.: Age differences in movement patterns used to rise from a bed in subjects in the third through fifth decades of age. Physical Therapy **73**(5), 300–309 (1993)
6. Fujiwara, K., Kanehiro, F., Kajita, S., Yokoi, K., Saito, H., Harada, K., Kaneko, K., Hirukawa, H.: The first human-size humanoid that can fall over safely and stand-up again. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 2, pp. 1920–1926 (2003)
7. Hirukawa, H., Kajita, S., Kanehiro, F., Kaneko, K., Isozumi, T.: The human-size humanoid robot that can walk lie down and get up. The International Journal of Robotics Research **24**(9), 755–769 (2005)
8. Janssen, W.G., Bussmann, H.B., Stam, H.J.: Determinants of the sit-to-stand movement: a review. Physical Therapy **82**(9), 866–879 (2002)
9. Kalisiak, M., van de Panne, M.: RRT-blossom: RRT with a local flood-fill behavior. In: IEEE International Conference on Robotics and Automation, pp. 1237–1242 (2006)
10. Kanehiro, F., Fujiwara, K., Hirukawa, H., Nakaoka, S., Morisawa, M.: Getting up motion planning using mahalanobis distance. In: ICRA, pp. 2540–2545 (2007)
11. Kovar, L., Gleicher, M., Pighin, F.: Motion graphs. In: SIGGRAPH, vol. 21, pp. 473–482 (2002)
12. Kuffner, J., Nishiwaki, K., Kagami, S., Inaba, M., Inoue, H.: Motion planning for humanoid robots. In: ISRR, pp. 365–374 (2003)
13. Kuffner, J.J., LaValle, S.M.: RRT-Connect: an efficient approach to single-query path planning. In: IEEE Internatinal Conference on Robotics and Automation, pp. 995–1001 (2000)
14. LaValle, S.M.: Rapidly-exploring random trees: a new tool for path planning. Tech. rep. TR 98-11, Computer Science Dept., Iowa State University (1998)
15. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. International Journal of Robotics Research **20**(5), 378–400 (2001)
16. Lee, J., Chai, J., Reitsma, P.S.A., Hodgins, J.K., Pollard, N.S.: Interactive control of avatars animated with human motion data. ACM Trans. Graph. **21**(3), 491–500 (2002)
17. Liu, C.K., Popović, Z.: Synthesis of complex dynamic character motion from simple animations. In: ACM SIGGRAPH, pp. 408–416 (2002)
18. Liu, L., Yin, K., van de Panne, M., Shao, T., Xu, W.: Sampling-based contact-rich motion control. In: ACM SIGGRAPH, pp. 1–10 (2010)
19. Macchietto, A., Zordan, V., Shelton, C.R.: Momentum control for balance. ACM Trans. Graph. **28**(3), 1–8 (2009)
20. McCoy, J.O., VanSant, A.F.: Movement patterns of adolescents rising from a bed. Physical Therapy **73**(3), 182–193 (1993)
21. Morimoto, J., Doya, K.: Reinforcement learning of dynamic motor sequence: learning to stand up. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, pp. 1721–1726 (1998)
22. Pratt, J., Torres, A., Dilworth, P., Pratt, G.: Virtual actuator control. In: IEEE International Conference on Intelligent Robots and Systems, pp. 1219–1226 (1996)
23. Safonova, A., Hodgins, J.K., Pollard, N.S.: Synthesizing physically realistic human motion in low-dimensional behavior-specific spaces. ACM Trans. Graph. **23**(3), 514–521 (2004)
24. Shiratori, T., Coley, B., Cham, R., Hodgins, J.K.: Simulating balance recovery responses to trips based on biomechanical principles. In: Symposium on Computer Animation, pp. 37–46 (2009)
25. Tsai, Y.Y., Lin, W.C., Cheng, K.B., Lee, J., Lee, T.Y.: Real-time physics-based 3d biped character animation using an inverted pendulum model. IEEE Transactions on Visualization and Computer Graphics **16**(2), 325–337 (2010)
26. Ulbrich, J., Raheja, A., Alexander, N.B.: Body positions used by healthy and frail older adults to rise from the floor. Journal of the American Geriatrics Society **48**(12), 1626–1632 (2000)
27. VanSant, A.F.: Age differences in movement patterns used by children to rise from a supine position to erect stance. Physical Therapy **68**(9), 1330–1338 (1988)
28. VanSant, A.F.: Rising from a supine position to erect stance: description of adult movement and a developmental hypothesis. Physical Therapy **68**(2), 185–192 (1988)
29. Witkin, A., Kass, M.: Spacetime constraints. In: ACM SIGGRAPH, pp. 159–168 (1988)
30. Yamane, K., Kuffner, J.J., Hodgins, J.K.: Synthesizing animations of human manipulation tasks. ACM Trans. Graph. **23**(3), 532–539 (2004)
31. Yamane, K., Nakamura, Y.: Dynamics filter—concept and implementation of online motion generator for human figures. IEEE Transactions on Robotics and Automation **19**(3), 421–432 (2003)
32. Zordan, V.B., Majkowska, A., Chiu, B., Fast, M.: Dynamic response for motion capture animation. ACM Trans. Graph. **24**(3), 697–701 (2005)

Wen-Chieh Lin received the B.S. and M.S. degrees in control engineering from the National Chiao Tung University, Hsinchu, Taiwan, in 1994 and 1996, respectively, and the Ph.D. degree in robotics from Carnegie Mellon University, Pittsburgh, in 2005. Since 2006, he has been with the Department of Computer Science and the Institute of Multimedia Engineering, National Chiao Tung University, as an assistant professor. His current research interests include computer graphics, computer animation, and computer vision. He is a member of the IEEE and the ACM.

Yi-Jheng Huang received the B.S. and M.S. degrees from the Department of Computer Science and Information Engineering, National Dong Hwa University, and the Institute of Network Engineering, National Chiao Tung University, Taiwan, in 2007 and 2009, respectively. She is interested in computer graphics, image processing, computer vision, and software engineering.