Full-Body Hybrid Motor Control for Reaching

Wenjia Huang, Mubbasir Kapadia, and Demetri Terzopoulos

University of California, Los Angeles

Abstract. In this paper, we present a full-body motor control mechanism that generates coordinated and diverse motion during a reaching action. Our framework animates the full human body (stretching arms, flexing of the spine, as well as stepping forward) to facilitate the desired end effector behavior. We propose a hierarchical control system for controlling the arms, spine, and legs of the articulated character and present a controller-scheduling algorithm for coordinating the sub-controllers. High-level parameters can be used to produce variation in the movements for specific reaching tasks. We demonstrate a wide set of behaviors such as stepping and squatting to reach low distant targets, twisting and swinging up to reach high lateral targets, and we show variation in the synthesized motions.

1 Introduction

A fundamental research problem in motion planning for virtual characters (and robots) is to control the body in order to achieve a specified target position. It occurs frequently in game scenarios when characters must reach to grasp an object. There are three issues that may need to be addressed in generating realistic reaching motion: (1) correctness of hand placement, (2) satisfaction of the constraints on the character's body, and (3) naturalness of the whole-body movement. An ideal reaching motion requires the character's hand to follow a smooth, collision-free trajectory from its current position to the target position while satisfying the constraints on the entire body. Inspired by prior work [8, 11, 16], we consider balancing, coordinating, and variation for reaching motion that takes into account the motion of the arms, spine, and legs of an anthropomorphic character.

In particular, we develop a hierarchical control strategy for controlling a virtual character. We use analytical IK to control the hands of the character, thus providing fast and accurate control of the end effectors (hands). A stepping motion controller, coupled with a non-deterministic state machine, controls the lower limbs to move the character towards a target that is beyond its reach. The spine controller ensures the balance of the character by accounting for its center of mass, and uses rotation decomposition for simple and robust control. Finally, we develop a novel controller-scheduling algorithm that generates the coordination strategies between multiple controllers.

2 Related Work

There has been a considerable amount of research both in robotics [3, 17] and in animation [6, 10, 13–15, 19] that addresses the problem of actuator control. Techniques like Rapidly-Exploring Random Trees [4, 10] and Probabilistic Road Maps [7] focus on the correctness of the generated motion, while data-driven methods [9, 22] animate natural-looking characters. However, controllers that generate animations that are natural, varied, and work in a wide variety of scenarios (dynamic environments) have yet to be realized. The work in [8] uses hierarchical control for gesture generation. Spine rotation decomposition strategies have been presented in [16]. The work in [5] uses key posture interpolation to generate body postures and IK for limb movement.

Inverse Kinematics is often used in conjunction with motion-capture data for character animation [9]. There are three popular methods to solve IK problems. Jacobian-based numerical methods can deal with arbitrary linkages, but they are computationally intensive. Cyclic Coordinate Descent (CCD) [12] is efficient, but it tends to favor the joints near the end-effector [18]. Analytical IK methods [5, 11] for body posture control have proven to be a good solution provided the number of joints is kept small. Prior research [5, 20] models human limbs as 3 joints and 7 DOF linkages. Using this model, an analytical solution to the IK problem can be derived accurately and efficiently, and it is the chosen method for this work.

Our work differs from prior work as follows: We use a simplified character model coupled with an analytical IK solver for efficiently controlling the character's limbs. Next, we employ a coordinated control strategy for reaching motion that takes into account spine as well as limb motion. Finally, the spine rotation in our framework is decomposed into swing and twist components, which can be controlled independently.

3 Character Representation

In this section, we describe the character model that we use in our work and the controllers used to animate the character. Section 3.1 outlines the character representation (character skeleton and mesh structure). Section 3.2 describes our hierarchical control structure and Sections 3.3, 3.4, and 3.5 detail the implementation of the arm, spine (including head), and leg controllers, respectively.

3.1 Character Model

Figure 1(a) illustrates the mesh and skeleton structure of our anthropomorphic character model. The model is composed of 16 separate meshes and it is equipped with an articulated skeleton of 21 bones with 43 degrees of freedom. The meshes are bound to the skeleton through rigid skinning. The character has 4 linkages—two arms and two legs, each consisting of 3 bones and 4 joints. The extra joint is at the lowest level of the joint hierarchy (e.g., the finger tip of the arm linkage) and has 0 DOFs. The remaining three joints—the base, mid, and end joint—have a

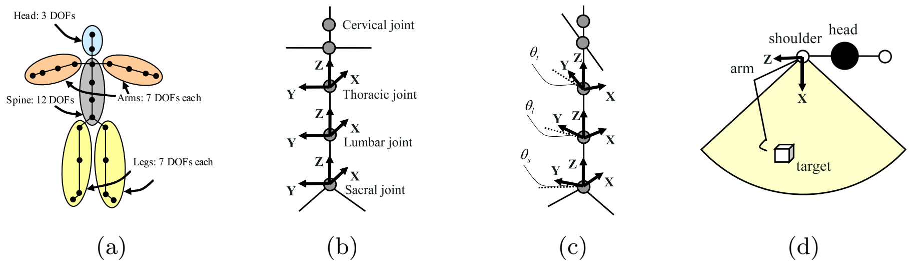

Fig. 1. (a) Character skeleton decomposition. (b) Spine modeling and local joint frames. (c) Twist rotations of all spine joints. (d) Top view of *comfortable reaching angle* (yellow). The y axis of the shoulder frame points out of the page.

total of 7 DOFs. The four linkages of the character are controlled using analytical inverse kinematics, as was described above.

The remaining joints model the spine of the character. Attempting to model the spine as a single bone would result in stiff and unnatural movement. The human spine comprises cervical, thoracic, lumbar, and sacral regions [2]. We therefore model the spine using four bones and four joints (cervical, thoracic, lumbar, and sacral) (Figure 1(b)). The spine's root joint (sacral) has 6 DOFs, with position as well as orientation control. The 2 intermediate spine joints (lumbar and thoracic) each have 3 DOFs. The neck joint (cervical) has 3 DOFs that serve to control the orientation of the character's head. Additional 0 DOF joints connect to the spine a shoulder bone, which provides attachment points for the arms, and a pelvic bone, which provides attachment points for the legs.

3.2 Hierarchical Control Structure

The character has a hierarchical skeletal structure. The spine is at the highest level of the hierarchy and it determines the position and orientation of the entire character. The head, arms, and legs are at the second level of the hierarchy. The arms and legs are controlled using analytical IK, facilitating easy control of the joint angles and specifying joint limits. The four controllers that we employ deal with different sets of hierarchical skeleton components. The arm controller enables the arm to perform a reaching motion. The spine controller controls the spine and decomposes rotations into swing and twist for simple and robust control. The head controller determines the direction in which the character's head is facing. The leg controller controls the legs and the root joint of the spine by generating stepping movements.

3.3 Arm Controller

Arm motion is a fundamental aspect of reaching, as it determines the ability and accuracy of the motion of the end-effector (hand) in reaching a desired position.

The end-effector is required to follow a smooth trajectory from its current position to the desired position, while obeying joint constraints and providing natural-looking motions (we do not deal with hand grasping motions). Modeling the trajectories of hand motion using Bezier curves [1] can yield satisfactory results, but this requires pre-captured hand motion data. We employ an ease-out strategy in which the hand trajectory is the shortest path to the target and the speed of the hand starts relatively high and gradually decreases as the hand reaches the target. This results in the hand slowing down naturally as it nears the target position and avoids "punching" the target.

The motion of the hand is governed by the starting velocity $\mathbf{v}_\mathrm{s}$, the deceleration $\mathbf{a}$, and the total time t_t. Given a start position $\mathbf{p}_\mathrm{s}$ and end position $\mathbf{p}_\mathrm{e}$, the target displacement is calculated as $\mathbf{d}_\mathrm{h} = \mathbf{p}_\mathrm{e} - \mathbf{p}_\mathrm{s}$. The motion of the hand has the following properties:

$$\mathbf{v}_\mathrm{s} = v_\mathrm{s}\frac{\mathbf{d}_\mathrm{h}}{|\mathbf{d}_\mathrm{h}|}, \tag{1}$$

$$\mathbf{a} = \frac{-\mathbf{v}_\mathrm{s}}{t_\mathrm{t}(1-r_\mathrm{s})}, \tag{2}$$

$$\mathbf{d}_\mathrm{h} = \mathbf{v}_\mathrm{s} t_\mathrm{t} + \frac{1}{2}\mathbf{a}(t_\mathrm{t}(1-r_\mathrm{s}))^2, \tag{3}$$

where r_s, the ratio of the duration of the start phase to the total duration of the reaching motion, is used to determine the deceleration of the hand as it nears the target. Substituting equations (1) and (2) into (3), the total time t_t can be calculated as

$$t_\mathrm{t} = \frac{2|\mathbf{d}_\mathrm{h}|}{v_\mathrm{s}(r_\mathrm{s}+1)}. \tag{4}$$

Algorithm 1 describes the arm controller logic to initialize and update the position of the hand over time. The user-defined parameters v_s and r_s can be modified in order to achieve variation in the results. We initialize $v_\mathrm{s} = 20$ cm/s and $r_\mathrm{s} = 0.8$.

3.4 Spine Controller

The spine of the character plays a crucial role during reaching as the character needs to bend and twist its upper body to reach targets at different positions. The spine controller determines the upper-body position and orientation of the character (six degrees of freedom). Our four-joint spine model enhances the control of the character, resulting in more fluid and natural animations.

Spine Rotation Decomposition. Previous work [5] introduced swing-and-twist orientation decomposition for arm joints, enabling simple and robust control. Inspired by this work, we similarly decompose the orientations of the spine joints. The advantage of this decomposition method is that it separates the spine motion into two primitive orientations, each of which can be governed by a separate control module in order to achieve robust control.

Algorithm 1. Arm controller

```
Procedure Initialize($\mathbf{p}_s$, $\mathbf{p}_e$, $v_s$, $r_s$)
    Input: $\mathbf{p}_s$: Start hand position
    Input: $\mathbf{p}_e$: Target hand position
    Input: $v_s$: Start hand speed
    Input: $r_s$: Deceleration ratio
    Output: $t_t$: Total time
    Output: $\mathbf{a}$: Deceleration
    Output: $\mathbf{v}_s$: Start hand velocity
    Output: $\mathbf{p}_c$: Current hand position
    Output: $t_c$: Current execution time

    $\mathbf{d}_h := \mathbf{p}_e - \mathbf{p}_s$
    $t_t := 2|\mathbf{d}_h|/v_s(r_s + 1)$
    $\mathbf{a} := -\mathbf{v}_s/t_t(1 - r_s)$
    $\mathbf{v}_s := v_s\mathbf{d}_h/|\mathbf{d}_h|$
    $\mathbf{p}_c := \mathbf{p}_s$
    $t_c := 0.0$

Procedure Update($\mathbf{v}_s$, $t_t$, $r_s$, $\mathbf{a}$, $\mathbf{p}_e$, $\mathbf{p}_c$, $t_c$, $\Delta t$)
    Input: $\mathbf{v}_s$: Start hand velocity
    Input: $t_t$: Total time
    Input: $r_s$: Deceleration ratio
    Input: $\mathbf{a}$: Deceleration
    Input: $\mathbf{p}_e$: Target hand position
    Input: $\mathbf{p}_c$: Current hand position
    Input: $t_c$: Current execution time
    Input: $\Delta t$: Time step per frame
    Output: $\mathbf{v}_c$: Updated hand velocity
    Output: $\mathbf{p}_c$: Updated hand position
    Output: $t_c$: Updated execution time

    if $|\mathbf{p}_e - \mathbf{p}_c| < \varepsilon$ then stop
    if $t_c < t_t r_s$ then $\mathbf{v}_c := \mathbf{v}_s$
    else $\mathbf{v}_c := \mathbf{v}_s + \mathbf{a}(t_c - t_t r_s)$
    $\mathbf{p}_c := \mathbf{p}_c + \mathbf{v}_c \Delta t$
    $t_c := t_c + \Delta t$
```

The twist is a rotation around the z axis of the joint frame (Figure 1(c)). For the body to twist by an angle θ, the twist angle is decomposed among the three spine joints as follows:

$$\theta = \theta_s + \theta_l + \theta_t, \tag{5}$$

where θ_s is the twist angle of the sacral (root) joint, and θ_l and θ_t are the twist angles for the lumbar and thoracic joints, respectively. The twist angle θ is uniformly distributed among the three spine joints by setting $\theta_s = \theta_l = \theta_t$.

A *comfortable reaching angle* is defined for deciding the total twist angle, shown in Figure 1(d). The current reaching angle is the angle between the x axis and $\mathbf{d}_p$, the projection of the reaching vector $\mathbf{d}$ on the x-z plane of the shoulder frame, where $\mathbf{d} = \mathbf{p}_e - \mathbf{p}_{sh}$ is the vector from the shoulder to target, and

$$\mathbf{d}_p = \frac{[d_x, 0, d_z]^T}{\sqrt{d_x^2 + d_z^2}}. \tag{6}$$

A search process determines the total twist angle for which the target falls within the *comfortable reaching angle*. This angle is further decomposed into rotations of each spine joint (Figure 1(c)). For the whole spine to twist by an angle θ, each of the three lower spine joints need to twist by an angle $\theta/3$.

The swing is a rotation around an axis that lies in the x-y plane of the joint frame (Figure 2(a)). The swing axis is taken to be the one that most efficiently lowers or raises the upper body towards the target for the arm to reach the target. A plane is defined by $\mathbf{d}$ and the normal $\mathbf{y}$ of the ground plane. The swing axis $\mathbf{s}$ is defined as the normal of this plane (Figure 2(b)), which ensures the efficient movement of the upper body in the vertical direction:

$$\mathbf{s} = \mathbf{d} \times \mathbf{y}. \tag{7}$$

A *start swing distance* is defined for determining the swing angle of each joint. If the current vertical distance between the shoulder and the target is larger than

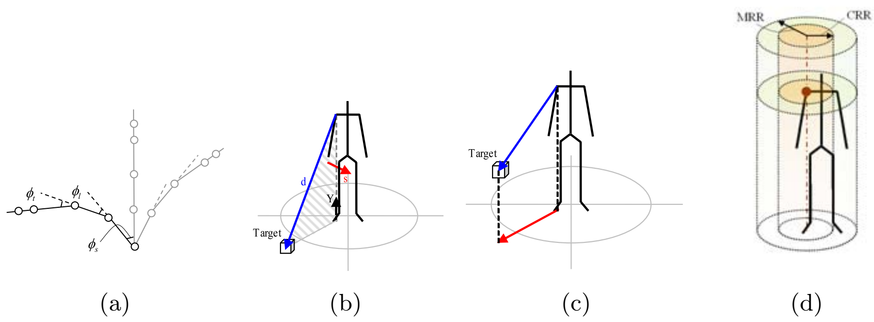

Fig. 2. (a) Swing rotation. (b) Swing axis. (c) Stepping direction (red) as the projection of arm reaching direction (blue) to ground. (d) Convenient reaching range CRR (inner cylinder) and maximum reaching range MRR (outer cylinder) of the character.

the start swing distance, the spine will swing. The swing angle ϕ is calculated by first determining the direction of the swing (clockwise or anti-clockwise) and then the three swing angles are iteratively increased until the hand is able to reach the target (Figure 2(a)).

Center of Mass Adjustment. A shift of the center of mass (COM) is observable during spine motion. Consider a reaching motion when bending the upper body down to pick up an object. In order to balance the character, the hip must move backwards in order to maintain the COM within the support polygon of the character. Similarly when reaching a high target, the hip must move forward to maintain balance.

In our framework, we adjust the position of the sacral joint, which determines the root of the spine, to generate visually appealing balancing motion during spine swing. We do not adopt kinetic methods like [11], thus achieving fast performance in spine motion synthesis. The root displacement is computed as a function of the swing angle ϕ and $\mathbf{d}_p$, projecting reach direction $\mathbf{d}$ on the x-z plane:

$$\mathbf{r} = -f(\phi)\mathbf{d}_p, \tag{8}$$

where we choose $f(\phi) = k\phi$. The constant $k = |\mathbf{r}_{\text{max}}|/\phi_{\text{max}}$ is the ratio of the maximal root displacement to the maximum swing angle. In our implementation, we set $\phi_{\text{max}} = 120$ deg and $|\mathbf{r}_{\text{max}}| = 0.1$ m.

Head Control. Intentional head motion through the control of the cervical joint of the spine can improve the liveliness of a character. The head rotates to face the target in order to help the eyes track the target. We model the head motion in two phases—a start phase and a tracking phase. During the start phase, if the head is not facing the target, the orientation of the head gradually adjusts until it aligns with the vector $\mathbf{d}_t = \mathbf{p}_{\text{e}} - \mathbf{p}_n$, where $\mathbf{p}_{\text{e}}$ is the position of the target and $\mathbf{p}_n$ is the position of the cervical joint. Given the start orientation and the target direction, the intermediate rotations are calculated by spherical linear interpolation. The speed of the head rotation is open to user specification. During the tracking phase, the head continually updates its orientation in order to track the updated target direction, which will vary due to the relative displacement between the head and the target.

3.5 Leg Controller

When synthesizing reaching motion, the focus tends to be on arm control and stepping motion is often neglected. Humans frequently step in the direction of the target in order to assist their reaching action. In doing so, the target falls within the convenient reaching range; i.e., it is neither too close nor too far from the actor. Some research has considered this topic in the context of motion planning [7]. Reference [23] considered stepping as a dynamic stability problem. We address the problem as an intentional behavior, employing a leg controller that uses a probabilistic state machine to determine the stepping motion of the virtual character.

Stepping Motions. Two kinds of stepping motions can occur during the reaching process—half stepping or full stepping. Half stepping involves only one foot step forward while the other remains at its original position. Full stepping involves both feet moving forward to assist the reaching motion. One can observe that when the step length is small, people usually perform half stepping, while a large step length often compels full stepping. However, this is also subject to individual "random" factors, which is considered in our implementation.

Two parameters govern stepping motion synthesis—step length and step direction. The direction of the step is determined by $\mathbf{d}_p$, the projection of the reaching direction $\mathbf{d}$ on the x-z plane, as shown in Figure 2(c). The step length is initialized to a empirically selected default value of 0.433 m, and it is adjusted in the stepping decision phase to produce stepping variations.

A half stepping motion comprises the movement of the spine root as well as one foot end-effector. The animation literature [21] describes the trajectory of the foot as a convex arc and the trajectory of the root as a concave arc. We simplify these two arcs as piecewise linear functions. Here, the other foot is constrained to stay in its original position.

The full stepping motion control is similar to that of half stepping, but it involves the movement of both feet. After the first foot lands on the ground, the other foot follows in the same direction. Meanwhile, the root moves slowly until the second foot reaches the spot of the first foot.

Stepping Decision State Machine. A probabilistic state machine is used to determine the stepping motion of a character that assists its reaching motion. The requirement and length of a step is determined by the *maximal reaching range (MRR)* and *convenient reaching range (CRR)* of the character, as shown in Figure 2(d). These are defined as two cylinders centered at the shoulder of the reaching arm. The stepping decision state machine works as follows:

- If the target is outside the *MRR* of the character, the character will perform a stepping motion.
- If the target falls between the *CRR* and *MRR* of the character, the character will step with a probability $P(\frac{d-d_c}{d_{\max}-d_c})$, where $d=\sqrt{d_x^2+d_z^2}$ is the horizontal distance of the target from the shoulder, d_c is the convenient reaching distance, and $d_{\max}$ is the maximum reaching distance.
- If the target falls within the *CRR*, the character will not perform a stepping motion.

If the character is to perform a stepping motion, the choice of half-stepping or full-stepping is determined based on the principle that a character is more likely to take a full step for targets farther away and will take a half step for nearby targets. The probability P_{half} that a character will take a half step is $P_{\text{half}} = P_{\max}$ when $d = d_c$ and it decreases linearly to $P_{\min}$ as d approaches d_{max}, where $d_c < d < d_{\max}$. Meanwhile, the probability of taking a full step, $P_{\text{full}} = 1 - P_{\text{half}}$.

The step length l is computed as a function of the horizontal distance of the target from the shoulder d and the maximum step length $l_{\max}$ and minimum step length $l_{\min}$ of the character, as follows:

$$l = \begin{cases} l_{\min} & \text{if } d \le d_c; \\ \frac{l_{\max}|d-d_c|p+l_{\min}|d-d_{\max}|(1-p)}{|d-d_c|p+|d-d_{\max}|(1-p)} & \text{if } d_c < d < d_{\max}; \\ l_{\max} & \text{if } d \ge d_{\max}, \end{cases} \tag{9}$$

where p is a normally distributed random number.

Leg Flexibility. The flexibility of the legs is an interesting motion feature that manifests itself during a reaching action. In order to pick up an object from the ground, the character needs to choose between stooping or squatting to lower its upper body. One factor in producing this motion is the flexibility of the legs. If the flexibility is low, a stoop is preferable in the reaching motion, otherwise if the flexibility is high, a squat is preferred. The squat motion involves lowering the position of the root of the spine, while the stoop motion does not. A stoop range is computed as the ratio of the length of the arm and a user-defined “flexibility parameter. If the vertical distance between the shoulder and the target exceeds this range, the root is triggered to squat down in order to reach the target.

4 Motion Controller Scheduling for Coordinated Reaching Motion

The order of movement of the arm, legs, and spine often varies with each situation and for different individuals. Hence, there exist multiple valid orderings of the individual control strategies that produce realistic animation. For example, a character may first choose to minimize the distance to the target in the horizontal space (stepping motion), then twist the body to achieve a comfortable reaching

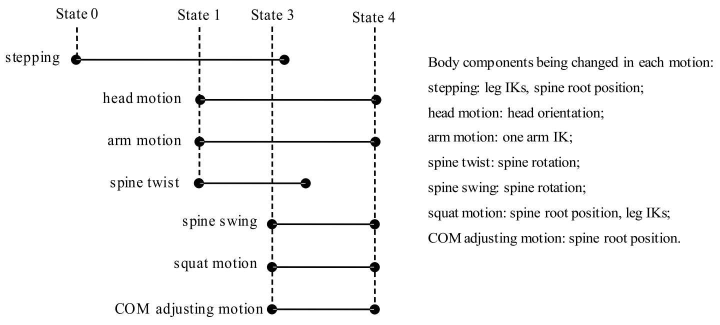
State 0
State 1
State 3
State 4
stepping
head motion
arm motion
spine twist
spine swing
squat motion
COM adjusting motion
Body components being changed in each motion:
stepping: leg IKs, spine root position;
head motion: head orientation;
arm motion: one arm IK;
spine twist: spine rotation;
spine swing: spine rotation;
squat motion: spine root position, leg IKs;
COM adjusting motion: spine root position.

Fig. 3. Motion controller scheduling

angle (spine twist rotation), then minimize the distance in the vertical space (spine swing rotation), and finally position the hand to touch the object. This ordering of controllers may easily be changed in a different situation.

To address the non-uniqueness, we devise a motion controller scheduling strategy that selects a sequence of one or more controllers to animate a virtual character for a particular reaching task. An overview of the motion controller scheduling algorithm is presented in Figure 3. First, we identify four states in which a character may find itself during any given reaching task. In each state, one or multiple controllers are triggered to perform stepping, arm, head, or spine motion in an effort to reach the target. These states are evaluated in sequence, implicitly ordering the selection of the controllers. The state definitions and associated controller selection strategies are as follows:

- **State 0.** Check if the target is between the *CRR* and *MRR* or outside the *MRR* of the character. If so, trigger the stepping motion controller, which determines the step direction, step length and type of step—half step or full step; then transition to State 1.
- **State 1.** Check if the character is not performing a stepping motion; i.e., the target is within the convenient reaching range or the stepping motion has progressed to a certain time t_{st}. If so, trigger the arm and head controllers and trigger the spine twist rotation if necessary; then transition to State 2.
- **State 2.** Check if the character is not performing a spine twist motion or the twisting has progressed to a certain time t_{tw}. If so, trigger the spine swing rotation along with the COM adjusting motion. Also, trigger the squat motion if necessary; then transition to State 3.
- **State 3.** Check if the arm has reached the target or the maximum time to reach the target is up. If so, stop all the controllers. The animation is considered complete.

The values of t_{st} and t_{tw} determine the transitions between States 1 and 2 and States 2 and 3. They serve as intuitive, user-defined parameters with which to produce varied motions. The user provides values of the ratios $r_{\mathrm{st}} = t_{\mathrm{st}}/t_{\mathrm{st}}^{\mathrm{total}}$

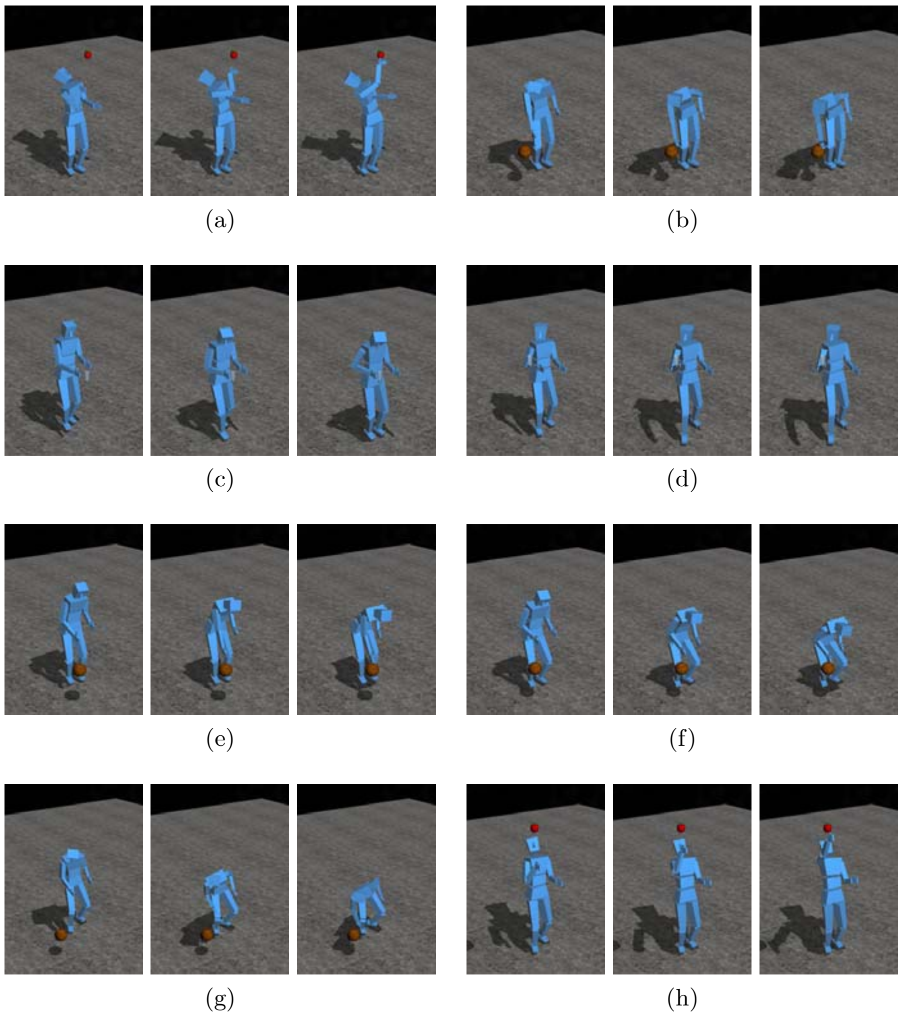

Fig. 4. A diverse set of reaching motions synthesized using our method. (a)–(b) Swing and twist spine motion: (a) The character reaches an apple to the upper-left; (b) the character reaches a basketball to the lower-right. (c)–(d) The character performs full stepping and half stepping to reach a glass. (e)–(f) The character with different leg flexibilities: (e) The character has a lower leg flexibility than in (f) as it stoops to reach a low target; (f) the character squats to reach the same low target. (g)–(h) Combined results: (g) The character steps and squats to reach a low basketball; (h) the character steps and swings up to reach a high apple.

and $r_{\mathrm{tw}} = t_{\mathrm{tw}}/t_{\mathrm{tw}}^{\mathrm{total}}$, from which t_{st} and t_{tw} are automatically computed as the points in time to transition to the next state. The scheduling algorithm can deal with most reaching tasks. For unreachable targets, the motion synthesis will stop as the hand arrives at the nearest position to the target or at time out.

5 Results

Figure 4 illustrates a diverse set of reaching motions synthesized for a full-body character using our method. For targets beyond its reach, the character chooses appropriate stepping motions and spine motions and it integrates them with head and hand motions to achieve coordinated, graceful whole-body movement. Diversity in the motion is achieved using a probabilistic model for stepping along with user defined parameters such as leg flexibility. Coordination and diversity are integrated in the motion synthesis by the modular design of the motion controllers with the systematic scheduling algorithm.

6 Conclusion and Future Work

We introduced a set of efficient motor controllers for the arms, spine, and legs of an anthropomorphic character and proposed a controller scheduling algorithm to coordinate the movements of these body parts in order to synthesize natural reaching motions. Our results demonstrate diverse reaching actions coordinated with spine and stepping motions.

More spine configurations and stepping with additional steps should be considered in future work. Like [22], motion capture data can be coupled with controllers to improve the naturalness of the synthesized motion. Moreover, we would like to extend our method to consider collision avoidance as well as to handle dynamic targets. A more intelligent control mechanism such as a planner would be required to handle space-time goals.

References

1. Faraway, J., Reed, M., Wang, J.: Modeling 3D trajectories using Bezier curves with application to hand motion. Applied Statistics 56, 571–585 (2007)
2. Gray, H.: Anatomy, Descriptive and Surgical. Gramercy, New York (1977)
3. Inoue, K., Yoshida, H., Arai, T., Mae, Y.: Mobile manipulation of humanoids: Real-time control based on manipulability and stability. In: Proc. IEEE Int. Conf. on Robotics and Automation, pp. 2217–2222 (2000)
4. Kallmann, M.: Scalable solutions for interactive virtual humans that can manipulate objects. In: Proc. 1st Conf. on Artificial Intelligence and Interactive Digital Entertainment, pp. 69–75 (2005)
5. Kallmann, M.: Analytical inverse kinematics with body posture control. Computer Animation and Virtual Worlds 19(2), 79–91 (2008)
6. Kallmann, M.: Autonomous object manipulation for virtual humans. In: SIGGRAPH 2008: ACM SIGGRAPH 2008 Courses, pp. 1–97. ACM, New York (2008)

7. Kallmann, M., Aubel, A., Abaci, T., Thalmann, D.: Planning collision-free reaching motions for interactive object manipulation and grasping. Computer Graphics Forum (Proc. Eurographics 2003) 22(3), 313–322 (2003)
8. Kallmann, M., Marsella, S.: Hierarchical motion controllers for real-time autonomous virtual humans. In: Panayiotopoulos, T., Gratch, J., Aylett, R.S., Ballin, D., Olivier, P., Rist, T. (eds.) IVA 2005. LNCS (LNAI), vol. 3661, pp. 243–265. Springer, Heidelberg (2005)
9. Kovar, L., Gleicher, M., Pighin, F.: Motion graphs. In: SIGGRAPH 2008: ACM SIGGRAPH 2008 Courses, pp. 1–10. ACM, New York (2008)
10. Kuffner Jr., J., Latombe, J.C.: Interactive manipulation planning for animated characters. In: Proc. Pacific Graphics, p. 417 (2000)
11. Kulpa, R., Multon, F.: Fast inverse kinematics and kinetics solver for human-like figures. In: 5th IEEE-RAS Int. Conf. on Humanoid Robots, pp. 38–43 (December 2005)
12. Kulpa, R., Multon, F., Arnaldi, B.: Morphology-independent representation of motions for interactive human-like animation. CG Forum 24, 343–352 (2005)
13. Lee, S.H., Sifakis, E., Terzopoulos, D.: Comprehensive biomechanical modeling and simulation of the upper body. ACM Trans. on Graphics 28(4), 99, 1–17 (2009)
14. Lee, S.H., Terzopoulos, D.: Heads up! Biomechanical modeling and neuromuscular control of the neck. ACM Transactions on Graphics 25(3), 1188–1198 (2006)
15. Liu, Y.: Interactive reach planning for animated characters using hardware acceleration. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA (2003)
16. Monheit, G., Badler, N.I.: A kinematic model of the human spine and torso. IEEE Computer Graphics and Applications 11(2), 29–38 (1991)
17. Murray, R.M., Sastry, S.S., Zexiang, L.: A Mathematical Introduction to Robotic Manipulation. CRC Press, Inc., Boca Raton (1994)
18. Park, D.: Inverse kinematics, `http://sites.google.com/site/diegopark2/computergraphics`
19. Shapiro, A., Kallmann, M., Faloutsos, P.: Interactive motion correction and object manipulation. In: Proc. Symp. on Int. 3D Graphics and Games, pp. 137–144 (2007)
20. Tolani, D., Goswami, A., Badler, N.I.: Real-time inverse kinematics techniques for anthropomorphic limbs. Graphical Models and Im. Proces. 62(5), 353–388 (2000)
21. Williams, R.: The Animator's Survival Kit. Faber, London (2001)
22. Yamane, K., Kuffner, J.J., Hodgins, J.K.: Synthesizing animations of human manipulation tasks. ACM Transactions on Graphics 23(3), 532–539 (2004)
23. Yoshida, E., Kanoun, O., Esteves, C., Laumond, J.P.: Task-driven support polygon humanoids. In: 6th IEEE-RAS Int. Conf. on Humanoid Robots (2006)