Terrain Runner: Control, Parameterization, Composition, and Planning for Highly Dynamic Motions

Libin Liu* KangKang Yin† Michiel van de Panne‡ Baining Guo§

* Tsinghua University † National University of Singapore ‡ University of British Columbia § Microsoft Research Asia

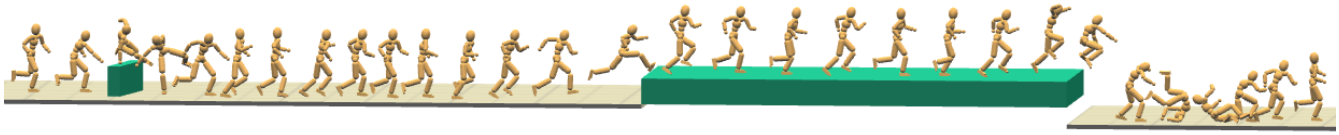

Figure 1: *An animated character runs, vaults, jumps, and drop-rolls across a parkour terrain during a real-time physics-based simulation. Given a single motion capture clip of each of these four skills as input, our method uses an offline process to develop robust control policies for parameterized versions of these skills, as well as robust transition motions.*

Abstract

In this paper we learn the skills required by real-time physics-based avatars to perform parkour-style fast terrain crossing using a mix of running, jumping, speed-vaulting, and drop-rolling. We begin with a single motion capture example of each skill and then learn reduced-order linear feedback control laws that provide robust execution of the motions during forward dynamic simulation. We then parameterize each skill with respect to the environment, such as the height of obstacles, or with respect to the task parameters, such as running speed and direction. We employ a continuation process to achieve the required parameterization of the motions and their affine feedback laws. The continuation method uses a predictor-corrector method based on radial basis functions. Lastly, we build control laws specific to the sequential composition of different skills, so that the simulated character can robustly transition to obstacle clearing maneuvers from running whenever obstacles are encountered. The learned transition skills work in tandem with a simple online step-based planning algorithm, and together they robustly guide the character to achieve a state that is well-suited for the chosen obstacle-clearing motion.

Keywords: physics-based animation, motion control, parkour

Links: DL PDF

1 Introduction

The physics-based animation of human-like characters has seen many significant advances, especially for locomotion tasks. However, the abilities and overall agility of the proposed control methods still fall far short of many human motions, such as that exhibited by a freerunner performing parkour skills [Edwardes 2009]. Parkour is a physical discipline which focuses on efficient movement around obstacles, and is considered as a sport, an art, or sometimes even a philosophy. The control, agility, expressivity, and versatility that human athletes demonstrate in highly dynamic motions such as parkour makes them mesmerizing to watch.

Reproducing the abilities of a high-level parkour athlete can be considered a "grand challenge" problem for character animation for several reasons. First, reference motions are difficult to capture. Capturing highly dynamic motions such as those used in parkour requires large space, special equipment, and trained athletes, in contrast to easy-to-capture motions such as walking. We thus desire a method that requires only a sparse set of motion capture examples. Second, designing robust motion controllers usually requires domain knowledge and human insight. However, the required insights are hard to obtain for unfamiliar and highly dynamic motions. A largely-automated method that is broadly applicable to a variety of motions is thus desirable. Third, the sequential composition of highly dynamic skills is difficult to develop using existing approaches for physics-based control. The end state resulting from one controller can easily fall outside the basin of attraction of the next controller, and the physical barriers in terrain-crossing tasks can appear at any distance with arbitrary height.

We present the first attempt at achieving the real-time physics-based simulation of an integrated subset of parkour skills, including running, jumping, speed vaulting, and drop-rolling. While this set of skills falls far short of a complete repertoire of parkour skills, we believe that it provides an important proof-of-concept that integrated sets of parameterized skills can be designed for highly dynamic motions. We present three primary contributions:

- Parameterization: Given a single motion-capture example clip for each skill, we develop closed-loop feedback controllers for parameterized versions of the motion. This entails learning *parameterized* low-dimensional feedback strategies and developing a multidimensional continuation method.
- Sequential Composition: We introduce a structured composition scheme for transitioning between running and the obstacle clearing maneuvers. The composition scheme is coupled to both the offline parameterization and the online planning. In this way the running steps used to approach an obstacle can be adjusted in the best way to achieve more realistic and

*e-mail:llb05@mails.tsinghua.edu.cn
†e-mail:kkyin@comp.nus.edu.sg
‡e-mail:van@cs.ubc.ca
§e-mail:bainguo@microsoft.com

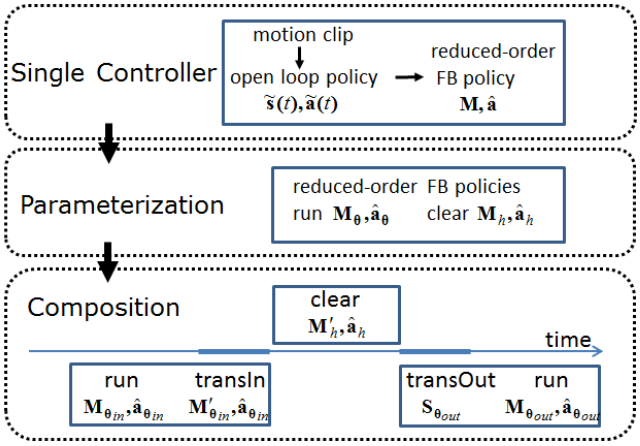

Figure 2: *System overview, excluding the online planning.*

robust obstacle clearing.

- System: The overall system provides insights into the representations and algorithms that can be used to develop parameterized and composable controllers for a set of highly dynamic motions, beginning from a single motion capture clip for each skill.

Figure 2 gives an overall system overview of the three offline stages involved in our clips-to-terrain-running-skills pipeline. First, a *single controller* is developed for each skill. This involves two steps: (a) given a single motion clip as input, reference trajectories are determined for the sensory state and actions that will define a baseline for open-loop control; and (b) learning a reduced-order linear feedback policy that then provides closed-loop control around these reference trajectories. Second, a *parameterization* process is used to transform the single controller instance obtained from the previous stage into a family of controllers that can produce motion variations, such as clearing obstacles of different heights, or running with a range of speeds and turning rates. Third, the *composition* of the skills is considered by developing controllers dedicated to the transition motions between running and the obstacle clearing skills. This involves the combined optimization of the affine feedback policy as well as the best choice of entry running speed and exit speed that yield robust transitions.

2 Related Work

There exists a large body of work related to the general problem we wish to solve. We only discuss the most related and recent publications here due to space constraints. For the clarity of presentation, we classify these papers into themes based on their principal contributions, even though specific papers may relate to several themes.

Kinematic Methods: Motion-graphs style techniques [Arikan and Forsyth 2003; Kovar et al. 2002; Lee et al. 2002; Kovar and Gleicher 2004] organize a corpus of motion capture data into graphs that encapsulate connections among segments of motion. Reinforcement learning extends motion graphs through the use of controllers that are trained offline, so that the best motions for a given task can be selected and interpolated online at interactive rates [McCann and Pollard 2007; Treuille et al. 2007; Lee et al. 2009; Lee et al. 2010a]. Parametric motions and deformable motions [Heck and Gleicher 2007; Min et al. 2009; Choi et al. 2011] have also been explored for interactive motion synthesis or navigation in cluttered virtual environments. These systems usually require rich motion examples, and do not guarantee physical realism, particularly for situations where the virtual character interacts with the environment through a multitude of contacts.

Physics-based Methods: Trajectory optimization can incorporate physics constraints, in addition to user constraints and statistical models [Safonova et al. 2004; Chai and Hodgins 2007; Wei et al. 2011]. Much research effort has been focused on developing feedback-based control strategies for locomotion tasks [Yin et al. 2007; da Silva et al. 2008; Muico et al. 2009; Coros et al. 2009; Coros et al. 2010; Lee et al. 2010b; Kwon and Hodgins 2010]. Controller optimization is also commonly used to further improve the capability and robustness of the basic locomotion controllers [Yin et al. 2008; Wang et al. 2009]. Locomotion control and planning have been tackled in an integrated fashion by planning with simplified dynamics or abstract models, and then optimizing for full-body trajectories to follow the low-dimensional plans [da Silva et al. 2008; Mordatch et al. 2010; Ye and Liu 2010]. Highly dynamic motions, such as handspring vaults [Hodgins et al. 1995] and snowboard stunts [Zhao and van de Panne 2005] have been studied also, but with a significant degree of manual design effort. To our knowledge, motions inspired by parkour, such as those investigated in this paper, have rarely been looked at.

Recently Liu et al. [2010] demonstrated automatic control reconstruction for arbitrary motions, including contact-rich motions. However, their reconstructed controls are open loop in nature and do not handle perturbations. Any changes in the environment or initial state require an additional pass of adaptation. We use the control reconstructions from Liu et al. as feed-forward reference controls to learn closed-loop feedback policies that are robust to run-time perturbations.

Control Composition: Research on the composition of control policies for heterogenous motions is relatively sparse. Direct parameter interpolation is sufficient for the transitions between various quadruped gaits [Coros et al. 2011]. For bipeds, the composable controllers of Faloutsos et al. [2001] learn an oracle that predicts the success or failure of a given controller as a function of the given current state. Another approach is to progressively populate a space of transition motions by developing new transition controls whenever unsuccessful transitions are encountered [Sok et al. 2007]. Linear Bellman combination [da Silva et al. 2009] can interpolate temporally aligned optimal policies for similar tasks with different boundary conditions. Composite controllers that track multiple trajectories in parallel have been demonstrated to improve robustness for locomotion tasks [Muico et al. 2011]. In our work, we systematically optimize for composition controls between running and parameterized obstacle clearing maneuvers, and in addition use online refinement of the running steps that approach obstacles.

3 Feedback Control from Single Example

We captured one example motion for each of four parkour skills: running, jumping, speed vaulting, and drop-rolling. Running is the main locomotion mode for fast terrain traversal, while the other skills are executed when the character encounters terrain variations or obstacles, such as a stair or a cliff. Hereafter we will refer to these skills as obstacle clearing maneuvers.

We use the sampling-based method of Liu et al. [2010] to construct open-loop controls from a single example of each of these actions. These controls are PD (Proportional Derivative) target-angle trajectories that are computed using an offline search procedure and therefore cannot adapt to external perturbations. In search for more robust closed-loop control policies, we use an optimization-based

approach to learn reduced-order affine feedbacks around the reference open-loop states $\tilde{\mathbf{s}}$ and actions $\tilde{\mathbf{a}}$. More specifically, we apply changes in control actions $\delta\mathbf{a}$ as an affine function of changes in sensory observations δs:

$$\delta\mathbf{a} = \mathbf{M_{ap}} \cdot \mathbf{M_{sp}} \cdot \delta\mathbf{s} + \hat{\mathbf{a}} \quad (1)$$

where

$$\delta\mathbf{a} = \mathbf{a} - \tilde{\mathbf{a}}, \delta\mathbf{s} = \mathbf{s} - \tilde{\mathbf{s}} \quad (2)$$

The affine term $\hat{\mathbf{a}} = \mathbf{0}$ for the default controllers constructed from motion examples, but will be nonzero later on during controller parameterization. In other words, the control laws are linear in this stage and affine in later stages. $\mathbf{M_{sp}}$ is an $r \times n$ sensory projection matrix that projects high-dimensional sensory observations to a reduced-order state space; and $\mathbf{M_{ap}}$ is an $m \times r$ action projection matrix that maps the reduced-order state back to the full action space. The reduced-order feedback policy has $r(m+n)$ parameters, and hereafter we will denote these parameters as $\mathbf{M}$ and simplify Equation 1 as:

$$\delta\mathbf{a} = \mathbf{M}\delta\mathbf{s} + \hat{\mathbf{a}} \quad (3)$$

We denote the full-body state as $\mathbf{s} = \{h_0, \mathbf{v_0}, \mathbf{q_0}, \dot{\mathbf{q}}_\mathbf{0}, \mathbf{q_j}, \dot{\mathbf{q}}_\mathbf{j}\}, j \in \{1, ..., k\}$, where h_0 is the height of the root; $\mathbf{v_0}, \mathbf{q_0}, \dot{\mathbf{q}}_\mathbf{0}$ are the linear velocity, the orientation, and the angular velocity of the root. $\mathbf{q_j}, \dot{\mathbf{q}}_\mathbf{j}$ are the rotation and angular velocity of joint j in its parent body's coordinate frame. k is the number of joints of the character. The full action set comprises the full-body target pose $\mathbf{a} = \{\mathbf{q_j}\}$. For our character model, $\mathbf{s}$ is 88 dimensions and $\mathbf{a}$ is 39 dimensions. Learning feedback policies on these high dimensional state and action sets is simply too slow and runs into the problem of overfitting. We thus manually choose low-dimensional action vector $\mathbf{a}$ and sensory vector $\mathbf{s}$ for each motor skill. We note that the remaining degrees of freedom are simply controlled using the open-loop control provided by the method of Liu et al. [2010].

Running: We rely on human insights to choose several key sensory properties and action parameters for running. $\mathbf{s} = \{\mathbf{q_0}, \mathbf{c}, \dot{\mathbf{c}}, \mathbf{d}\}$, where $\mathbf{q_0}$ is the root orientation; $\mathbf{c}$ and $\dot{\mathbf{c}}$ are the CoM (Center of Mass) position and linear velocity; and $\mathbf{d}$ is the 3D vector pointing from the CoM to the stance foot. We choose the hips and the waist as key joints for action: $\mathbf{a} = \{\mathbf{q}_{swhip}, \mathbf{q}_{sthip}, \mathbf{q}_{waist}\}$. To achieve more coordinated spinal postures, $\delta\mathbf{q}_{waist}$is also applied to the chest joint. The manually-chosen state and action vectors are 12 and 9 dimensions, respectively.

We use a stochastic global optimization technique, Covariance Matrix Adaption (CMA) [Hansen 2006], to optimize the affine feedback structure in Equation 3. The optimization begins from an initial guess consisting of zero entries. For running control, the optimization is challenging due to the requirement for balance and the complexity of the dynamical system. We therefore break the optimization into multiple stages, each with increasing difficulty, and each using the solution of the previous stage as a starting point. Due to limited space, we refer interested readers to [Ding et al. 2012] for details on the learning process for the running feedback control.

Jumping: Jumping shares the same action set as running. The sensory variables for jumping, however, additionally include the position of the swing foot with respect to the root, for better control of the landing location on the higher terrain.

Vaulting: Vaulting is a more complex movement skill than jumping, and a single time-invariant feedback matrix for the full course of the maneuver is inadequate to produce robust and realistic controls throughout the motion. Following the common choice of motion decomposition developed by parkour experts [Edwardes 2009], we segment a full vault into three phases: phase 1 - one hand reaches out for the top of the obstacle, and both legs lift off from the ground; phase 2 - the body passes over the obstacle and one foot lands on the other side of the barrier; phase 3 - the character continues to move fluidly in the direction. Phase 1 starts from the first frame and ends when the CoM passes the hand-obstacle contact; phase 2 then starts until one foot touches the ground; phase 3 continues further until the end of the vault. The multi-phase feedback policy for vaulting then becomes $\delta\mathbf{a} = \mathbf{M_k}\delta\mathbf{s} + \hat{\mathbf{a}}_\mathbf{k}$, where $k \in \{1, 2, 3\}$ is the phase index that correlates with the character state and simulation time. The number of parameters of the reduced-order feedback matrices is $r \times (m + n)$ for single phase motions such as jumping, but is $r \times (m + \sum n_k)$ for multi-phase motions such as vaulting. Interested readers can preview Table 1 for the exact dimensionality of the optimization for each skill.

The sensory/action sets for vaulting are the same as those for jumping, except that during phase 1 we augment the sensory state with the position of the supporting hand in order to allow for better control of the hand-obstacle contact location, as will be detailed later (see Equation 11).

Drop-Rolling: We segment a drop-roll into four phases: phase 1 - the character lifts off from the higher terrain; phase 2 - the character jumps down to the lower terrain; phase 3 - the character rolls on the lower terrain; phase 4 - the character stands up and continues to move fluidly in the direction. Phase 1 starts from the beginning and ends when both feet leave the ground; phase 2 then starts until both feet land on the lower terrain; phase 3 rolls and ends when the CoM passes over the kneeling lower leg; phase 4 continues further to stand up until the end of the motion. We augment the basic running sensory set to include the left and right foot positions with respect to CoM during phase 2 and phase 3, for better control of the landing position in drop-rolling.

For all the clearing maneuvers, we again use CMA to optimize for the affine feedback policy. As these skills are short in time and non-repetitive, the feedback terms are very easy to optimize for. A couple of CMA iterations, with goals similar to Equation 10 are already sufficient.

4 Parameterization

The above closed-loop controllers are learned from a single example motion for each skill, and thus only work for running styles and obstacles that are close to those of the example motions. For expressive terrain crossing, parameterized feedback controllers that work for a large range of parameters are desirable. More specifically, we wish to achieve running at various speeds and turning rates, and obstacle clearing maneuvers over barriers of varying heights.

4.1 Running Parameterization

For running, we consider two dimensions for parameterization: $\boldsymbol{\theta} = (\phi, v)$, where ϕ is the turning angle per step and v is the running speed. From the above learned running policy at a default speed with no turning, we wish to search for a series of parameterized closed-loop affine controllers of the following form:

$$\delta\mathbf{a} = \mathbf{M}_{\boldsymbol{\theta}}\delta\mathbf{s} + \hat{\mathbf{a}}_{\boldsymbol{\theta}} \quad (4)$$

4.1.1 Action Set Augmentation

The action set $\mathbf{a}$ for the default running is too restrictive for the parameterization task now. For example, it is symmetric in nature and apparently a character cannot turn with a symmetric controller. We

thus augment the action set as $\mathbf{a} = \{\mathbf{a_l}, \mathbf{a_r}, \alpha, \beta\}$, where $\mathbf{a_l}$ and $\mathbf{a_r}$ are the same as the action set before, but are only applied either for the left-stance steps or the right-stance steps to generate asymmetric gait for turning. α and β are scaling parameters in space and time, respectively, to facilitate the synthesis of natural running in different speed. The augmented action set is a 20-dimensional vector.

The parameter α is applied to the full-body reference state and reference action. It is mainly for scaling the spatial range of motion, hinting that joints in slower running tend to rotate less. We apply α for positions and velocities as follows:

$$\mathbf{x} = \alpha(\mathbf{x} - \bar{\mathbf{x}}) + \bar{\mathbf{x}} \qquad \mathbf{x} \in \{h_0, \mathbf{q_0}, \mathbf{q_j}\} \tag{5}$$

$$\dot{\mathbf{x}} = \alpha\dot{\mathbf{x}} \qquad \dot{\mathbf{x}} \in \{\mathbf{v_0}, \dot{\mathbf{q}}_0, \dot{\mathbf{q}}_j\} \tag{6}$$

$\bar{\mathbf{x}}$ indicates the mean of a quantity in the reference trajectory, except that for the root orientation we use zero mean in favor of an upright trunk for the running character. Note that for rotations represented in quaternions, only the rotation angles are scaled and the rotation axes remain the same. This is equivalent to a SLERP (Spherical Linear Interpolation) interpolation with the Identity quaternion.

The parameter β linearly scales the duration of the reference trajectory, so that the desired motion can reference a sped-up or slowed-down version of the reference. In effect β adapts the running pace with respect to the running speed. The above model for scaling the reference motion amplitude and timing significantly improves the quality and extends the range of parameterization from the default controllers, as our results in Table 1 show.

4.1.2 Continuation

When the character runs forward at the default speed $\boldsymbol{\theta_0}$, $\mathbf{M}_{\boldsymbol{\theta_0}}$ in Equation 4 is exactly the feedback matrix found in Section 3, and $\hat{\mathbf{a}}_{\boldsymbol{\theta_0}} = \mathbf{0}$. From this initial closed-loop running, we can use continuation methods [Yin et al. 2008] to explore the parameter space $\boldsymbol{\theta}$ and generate a series of feedback controls for running with various turning rates at various speeds. More specifically, we can sample a series of running speed $\{v_k\}$, which collectively span the target speed range, with small and gradual increments or decrements, as shown in Figure 3(a). Then from each v_k we sample along the axis of turning angle ϕ as in Figure 3(b).

The output of the above continuation process is a collection of affine policies $\mathbf{M}_{\boldsymbol{\theta_k}}$ and $\hat{\mathbf{a}}_{\boldsymbol{\theta_k}}$. We can then use Radial Basis Functions (RBFs), as schematically shown in Figure 3(c), to fit to the controls at the sample points (RBF centers). The controls for any $\boldsymbol{\theta}$ can then be efficiently computed on demand. We use the smoothest interpolant RBF [Carr et al. 2001] as follows:

$$f(\boldsymbol{\theta}) = p(\boldsymbol{\theta}) + \sum_{i=1}^{N} \lambda_i \varphi(|\boldsymbol{\theta} - \boldsymbol{\theta}_i|) \tag{7}$$

where p is a polynomial of low degree. We use 1-degree polynomial $p(\boldsymbol{\theta}) = c_1 + c_2\phi + c_3 v$. φ is the basis function with many choices. Basis functions with non-compact support, such as the thin plate spline and the biharmonic spline, are better suited to extrapolation than those with compact support, such as Gaussians. We thus choose the biharmonic spline $\varphi(r) = r$ for its simplicity and good extrapolation quality.

The continuation and interpolation process as schematically shown from Figures 3(a) to (c) is straightforward, but can be improved significantly if we utilize the extrapolation capability of the RBF. More specifically, in the first iteration we choose to optimize controls for a few samples in the parameter space, as shown by the green dots in Figure 3(d). We then immediately fit an RBF to these

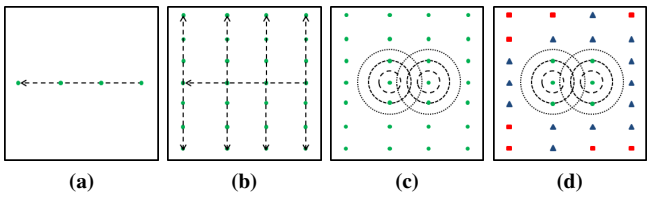

Figure 3: *Two dimensional continuation: (a) continuation along one axis first; (b) continuation along the other axis; (c) RBF interpolant to cover the full parameter space. (d) alternating extrapolation and continuation in the plane. Green dots are initial working samples; blue triangles are extrapolated samples that work; and red squares are the ones that fail.*

samples and extrapolate to the space surrounding the initial samples. Some nearby samples, represented by the blue triangles, are successful in the sense that the parkour avatar can accomplish the corresponding maneuver under the affine control generated by the RBF model. More distant samples, represented by the red squares in the figure, are more likely to fail. We select the outer boundary of the blue triangles for further optimization based on the initial RBF prediction of the controls. These improved blue triangles are then marked to become part of the set of 'solved' samples (green circles). The continuation process then repeats, i.e., fitting a new RBF to all solved examples, extrapolating, and optimizing the boundary, until we have fully covered the desired parameter space or no further improvements can be achieved.

The above predictor-corrector method implements a multi-dimensional continuation process that can cover a quasi rectangular space of [-6, 6]° × [2.0, 5.0]m/s within 11 hours on a multi-core computer. Here a quasi rectangle means there are a few failed samples near the boundary of the rectangular parameter space (Figure 6). As compared to a naive one-dimensional continuation process, utilizing the prediction power of the RBF model gains a speed-up of more than two folds. Note that in actual implementation we only need to optimize for the [0, 6]° region of the direction parameter space. In other words, we only optimize for controls for left turns, and then mirror the controls to get right turns. We also enforce control symmetry for straight line running with $\phi = 0$.

4.1.3 Optimization

For some chosen continuation parameters $\boldsymbol{\theta^*} = (\phi^*, v^*)$, for example the initial samples and the boundary blue triangles in Figure 3, we need to optimize the controls either from the default zero or from the controls derived from the latest RBF model. The optimization objective contains terms that penalize early failure (defined as the character falling or an unintended contact with the obstacle), excessive head movement, and deviations from the desired direction, speed, and stepping frequency. These are defined as:

$$E = w_t(N_d T_c - T_s) + \frac{w_h}{T_s}\int ||\mathbf{d}_h - \bar{\mathbf{d}}_h||dt$$
$$+ \frac{1}{N_s}\left(w_\phi|\phi_i - \phi^*| + w_v|v_i - v^*| + w_f|f_i - f^*|\right) \tag{8}$$

where $N_d = 20$ is the desired number of running cycles, T_c is the duration of one running cycle in the reference. The simulation is terminated once the character falls or when the desired number of cycles is reached. N_s is the actual number of cycles of the simulation, whose termination time is denoted as T_s. The second term stabilizes head movement, where $\mathbf{d}_h$ is the head position with respect to CoM in the character's facing frame, and $\bar{\mathbf{d}}_h$ is its average position.

ϕ_i, v_i and f_i are the turning angle, the speed and the stepping frequency of the stride cycle i, while ϕ^*, v^* and f^* are their desired values. Note that the continuation parameter $\boldsymbol{\theta^*}$ only consists of (ϕ^*, v^*). We estimate f^* heuristically from the desired speed: $(f^*/c_f) = 0.2(v^*/c_v) + 0.8$. This relationship is to guide the running pace to change in accordance with the speed, commonly observable from natural human running [Gutmann et al. 2006]. Without enforcing such a relationship, we can still succeed in optimizing running of different speed, but the motion will look like a slow motion of the same running for example, rather than a natural running motion where the pace adapts to the speed accordingly. Adding the space and time scaling parameters α and β into the action set alone, as described in Section 4.1.1, does not eliminate this problem completely. We therefore designate an estimated stepping frequency f^* to guide the optimizer towards desirable solutions. We approximate the normalization coefficients c_f and c_v by the average stepping frequency and speed of the captured example running, i.e., $c_f = 4.2$ steps/s and $c_v = 4.4$ m/s.

4.2 Clearing Maneuver Parameterization

For obstacle clearing skills, we consider the height of the obstacle h for parameterization:

$$\delta\mathbf{a} = \mathbf{M}_h \delta\mathbf{s} + \hat{\mathbf{a}}_h \tag{9}$$

Different from running parameterization, optimizing only for the control bias term $\hat{\mathbf{a}}_h$ is already enough to generate parameterized clearing maneuvers. So we leave $\mathbf{M}_h$ fixed at the value of the default controller until later in the composition stage of Section 5.1.

In addition to optimizing for the affine controls, we also optimize two additional parameters: u_h is the distance from the stance foot to the obstacle at the beginning of the maneuver; and v_h is the desired initial speed. The optimized u_h will later be used in motion composition and planning so that the character can plan to clear an obstacle of height h from an optimal distance. v_h will later be used to initialize the running speed that leads to the clearing maneuver during the optimization for motion composition. These two parameters enable more natural maneuvers that start further away from the obstacle with faster speed for clearing higher obstacles.

Similar to running parameterization, we need to augment the action set with the scaling parameters α and β to parameterize jumping and vaulting across a large range of obstacle height. Drop-rolling, however, does not need the spatial scaling parameter α. We also utilize the predictor-corrector method for continuation within the parameter space, but the RBF model is one-dimensional for clearing maneuvers instead of two-dimensional for running.

4.2.1 Optimization

The cost functions for optimizing clearing maneuvers for a chosen parameter share a common structure as follows:

$$E_h = w_c E_c + w_b E_b + w_p E_p \tag{10}$$

E_c controls desired contacts and penalizes unwanted collisions. It is the most important term for guiding clearing maneuvers to overcome obstacles. Different maneuvers use different E_c forms, which are generally intuitive and will be explained shortly. E_b measures how balanced the end state is. A clearing maneuver cannot be considered fully successful if the character falls after crossing the obstacle. We use $E_b = ||\mathbf{d} - \tilde{\mathbf{d}}||$ to compare the stance foot position to its desired position in the reference, both computed with respect to CoM. The term $E_p = \frac{w}{T}\int d_p(\mathbf{p}, \tilde{\mathbf{p}}) + d_p(\mathbf{p}_e, \tilde{\mathbf{p}}_e) + d_r(\mathbf{r}_e, \tilde{\mathbf{r}}_e)$ measures the difference between the output motion and the reference motion. The distance functions d_p and d_r measure the pose and root difference from their references, respectively. The latter two terms weigh the end state difference more to facilitate the transition from clearing back to running.

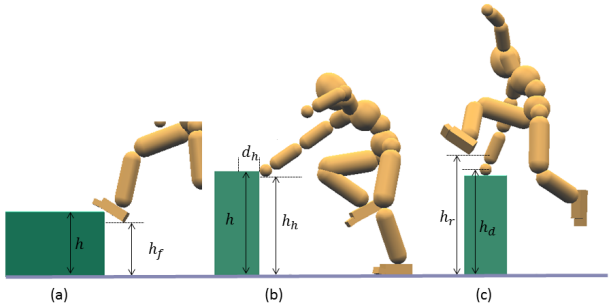

Figure 4: *Parameters illustration for optimizing obstacle clearing skills: (a) jumping; (b) vaulting phase 1; (c) vaulting phase 2.*

Jumping : We use contact control term $E_c = |h - h_f|_+$ for jumping. Here h_f is the lowest height of the contact foot (see Figure 4(a)) when it hits a stair of height h. The one-sided hat function $|x|_+$ is defined as $|x|_+ = max(x, 0)$. It penalizes the difference between the height of the foot and the height of the obstacle when the foot is below the obstacle. This term shapes the optimization to stay away from undesired collisions.

Vaulting: The contact control term for vaulting is defined as:

$$\begin{aligned} E_c = w_{f1}(|h_d - h_l|_+ + |h_d - h_r|_+) + w_{f2}||\mathbf{d}_r - \tilde{\mathbf{d}}_r|| \\ + w_{h1}(|h - h_h|_+ + d_h) + w_{h2}||\mathbf{d}_h - \tilde{\mathbf{d}}_h|| \end{aligned} \tag{11}$$

The first term guides both feet to overpass the obstacle. h_l and h_r are the lowest height of the feet when they pass the obstacle, and h_d is the desired minimal height for obstacle clearance (Figure 4(c)). The second term controls the contact position of the first landing foot, i.e., the right foot. $\mathbf{d}_r$ is the vector from the right foot to CoM. The third term guides the optimizer to put the hand onto the center of the obstacle for support of the full body. h is the height of the obstacle and h_h the lowest height of the hand when it contacts with the obstacle. d_h is the distance between the hand and the center of the obstacle (Figure 4(b)). The last term controls the hand position $\mathbf{d}_h$ with respect to CoM to stay close to the reference, because the hand supports the body when both feet are in the air similar to what a stance foot does during normal locomotion.

To vault high obstacles, the stance knee and shoulder are also key to generating the necessary thrust and support. Therefore we add these joints to the action set to easily discover control laws to overcome high barriers. In addition, our character's spherical hand representation cannot model the capability of firm grasping of human hands when in contact with the obstacle. We therefore treat the hand-obstacle contact as a ball-and-socket joint from the instance when contact is first established until when the character's CoM passes the contact location, which corresponds to the end of phase 1 for vaulting.

Drop-rolling: Drop-rolling usually does not involve unexpected collisions and therefore there is no E_c term. However, rotation and balance need to be controlled more precisely for drop-roll. We use an E_b term that is a weighted sum of the angular momentum deviation from the reference at the end of phase 1, the feet to CoM deviations at the end of phase 2, and the stance foot to CoM deviation at the end of phase 3: $E_b = w_L||\mathbf{L} - \tilde{\mathbf{L}}|| + w_d(||\mathbf{d}_l - \tilde{\mathbf{d}}_l|| + ||\mathbf{d}_r - \tilde{\mathbf{d}}_r|| + ||\mathbf{d} - \tilde{\mathbf{d}}||)$.

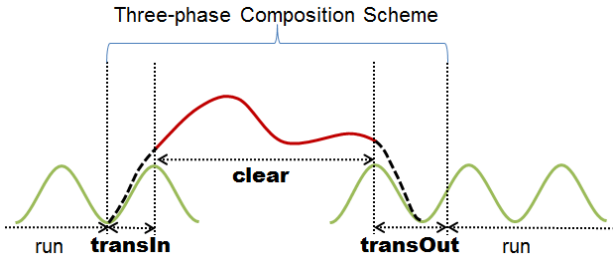

Figure 5: *Three-phase composition scheme and control blending. The solid lines are the control quantities for the individual skills, and the dashed lines are the blended quantities for transitions.*

5 Composition

To achieve parkour-style efficient movement around obstacles through a series of fast running and obstacle clearing maneuvers, the above individually developed parameterized controllers for different skills need to be composited dynamically for automatic terrain crossing. The highly dynamic nature of parkour motions means that the transitions in-to and out-of the parkour stunts are more challenging than for mundane tasks. In addition, our reference motions were captured from different subjects at different times, and are not even kinematically continuous. We therefore utilize a structured feedback policy search, as shown in Figure 5, for the best composition scheme: **transIn** – to transition running to clearing; **clear** – the actual obstacle clearing motion; **transOut** – to exit clearing motion and continue with running.

5.1 Three-phase Composition Scheme

Suppose we start from a running with parameter $\boldsymbol{\theta}_{in}$, which follows the control law $\delta\mathbf{a} = \mathbf{M}_{\boldsymbol{\theta}_{in}}\delta\mathbf{s} + \hat{\mathbf{a}}_{\boldsymbol{\theta}_{in}}$. Then when approaching an obstacle, we directly switch to the clearing maneuver of parameter h, which follows the control law $\delta\mathbf{a} = \mathbf{M}_h\delta\mathbf{s} + \hat{\mathbf{a}}_h$. Such direct transitions, however, usually fail because the discrepancy $\delta\mathbf{s}$ between the end state of the running and the start state of the clearing maneuver is too large for the affine feedback laws to handle. We therefore employ control blending, which will be discussed shortly in Section 5.3, and further optimize the controls of the last running step and the controls of the clearing maneuver as follows:

$$\delta\mathbf{a} = \mathbf{M}'_{\boldsymbol{\theta}_{in}}\delta\mathbf{s} + \hat{\mathbf{a}}_{\boldsymbol{\theta}_{in}} \tag{12}$$

$$\delta\mathbf{a} = \mathbf{M}'_h\delta\mathbf{s} + \hat{\mathbf{a}}_h \tag{13}$$

where $\mathbf{M}'_{\boldsymbol{\theta}_{in}}$ and $\mathbf{M}'_h$ can incorporate the necessary adjustments to achieve a successful transition. We initialize them with $\mathbf{M}_{\boldsymbol{\theta}_{in}}$ and $\mathbf{M}_h$, which are known after the previous parameterization step. We assume the running preparation of Equation 12 can be done within one running step as labeled by the **transIn** segment in Figure 5. After this preparation running step, then Equation 13 takes over as labeled by the **clear** segment in Figure 5.

Once the obstacle clearing movements finish, the character needs to regain balance and continue to run. Directly switching to a running with parameter $\boldsymbol{\theta}_{out}$ again usually fails. We thus execute a SIMBICON-style running controller $\delta\mathbf{a} = \mathbf{S}\delta\mathbf{s}$ during the **transOut** phase of Figure 5 to regain balance. The SIMBICON-style linear running controller has lower dimensions in both the sensory set and the action set [Yin et al. 2007], compared to the affine running controller we developed in Section 4.1. More specifically, $\mathbf{s} = \{\dot{\mathbf{c}}, \mathbf{d}_{st}, \mathbf{d}_{sw}\}$ is six dimensional, and $\mathbf{a} = \{q_x, q_z\}$ is two dimensional. $\dot{\mathbf{c}}$ is the planar velocity of the CoM, and $\mathbf{d}_{st}$ and $\mathbf{d}_{sw}$ are the planar vectors from the CoM to the stance and swing foot, respectively. q_x and q_z are the swing hip rotations around the horizontal axes. The SIMBICON-style running controller uses the reference motion of the affine running controller of the same speed as its reference trajectory as well, but it targets mainly on balance control. After the balance has been regained and the running has been stabilized, we then switch to the regular affine running controller with parameter $\boldsymbol{\theta}_{out}$.

To summarize, the free parameters for the three-phase composition scheme include:$(\boldsymbol{\theta}_{in}, \mathbf{M}'_{\boldsymbol{\theta}_{in}}, \mathbf{M}'_h, \mathbf{S}_{\boldsymbol{\theta}_{out}})$. $\boldsymbol{\theta}_{in}$ is the entering running velocity and angle. We always use zero turning angle as the entering direction, and initialize the velocity component with v_h found in the parameterization step for clearing maneuvers. This three-phase composition scheme is coupled with not only the previous motion parameterization step, but also the subsequent motion planning step. The best running parameters $\boldsymbol{\theta}_{in}$ leading up to the obstacle-clearing maneuvers will later inform the planner so that for an obstacle of a given height, the preparation running steps can be adjusted to the proper speed for more realistic and robust obstacle clearing. Interested readers can preview the exact dimensionality of the optimization problem for each skill in Table 1.

5.2 Optimization

We again use CMA to optimize for the composition policy. We first only optimize $(\boldsymbol{\theta}_{in}, \mathbf{M}'_{\boldsymbol{\theta}_{in}}, \mathbf{M}'_h)$ for the first two phases **transIn** and **clear**. For each simulation trial, i.e., each CMA sample, the character first runs at parameter $\boldsymbol{\theta}_{in}$ for two steps, then it executes **transIn** and **clear** and stops at the end of the clearing motion. The cost function of the optimization is the same as Equation 10, which measures the success of the clearing maneuver.

We run the CMA optimization until convergence. Then we reset the reference motion using the simulated motion of the converged controls, and run the CMA optimization for a second round to further enhance the robustness of the composition controls. For the second optimization stage, we also sample the end states of a number of runs around parameter $\boldsymbol{\theta}_{in}$ and use these as perturbed test states to further optimize the controls obtained from the initial optimization. After the CMA optimization converges again, we sample the end states of the clearing maneuvers and use their average speed as the exiting running velocity and zero as the exiting angle for $\boldsymbol{\theta}_{out}$.

We then optimize for $\mathbf{S}_{\boldsymbol{\theta}_{out}}$ in the **transOut** phase. A SIMBICON-style and a regular style running controller of parameter $\boldsymbol{\theta}_{out}$ are executed for a total of ten steps. The cost function of this optimization problem measures the success of the running:

$$E = w_t(N_dT_c - T_s) + \frac{w_h}{T_s}\int ||\mathbf{d}_h - \bar{\mathbf{d}}_h||dt + \frac{w_s}{T_s}\int d_p(\mathbf{s}, \tilde{\mathbf{s}})dt \tag{14}$$

The first two terms are the same as those of Equation 8. The third term measures the convergence to the running reference.

We reiterate that the above optimization targets a specific obstacle height h. We need to perform a continuation process similar to Section 4.1.2 to be able to compose running and clearing maneuvers to cross obstacles of different heights.

5.3 Reference Control Blending

Blending, or the idea of smoothing discontinuities within a time window, are commonly used for kinematically transition between motion examples [Arikan and Forsyth 2003; Kovar et al. 2002; Lee et al. 2002]. In our case when transitioning between different motion skills and control policies, discontinuities occur in the reference states $\tilde{\mathbf{s}}$ and actions $\tilde{\mathbf{a}}$. As a result, simulated motions can be

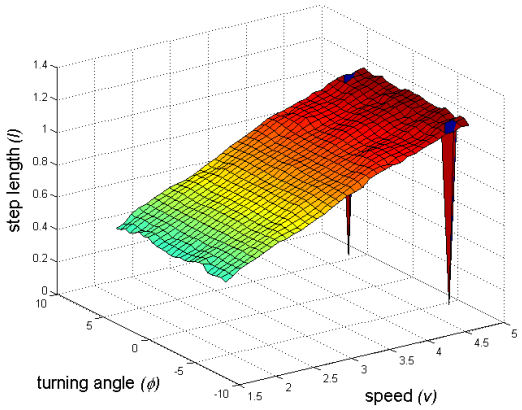

Figure 6: *The step length l with respect to the running parameter $\boldsymbol{\theta} = (\phi, v)$. The two outliers correspond to high-speed runs for which the optimization failed during parameterization.*

jerky around the transitions, and sometimes even fail the desired tasks, e.g., failing to vault over an obstacle. We therefore blend the reference motion $\tilde{\mathbf{s}}$ during the **transIn** and **transOut** segments of the composition for one running step, as illustrated in Figure 5. We can also choose to blend $\tilde{\mathbf{a}}$ as well. However, the reconstructed reference controls from Liu et al. [2010] are PD target-angle trajectories that offset the reference motion $\tilde{\mathbf{s}}$. Hence as long as we re-calculate the reference actions $\tilde{\mathbf{a}}$ from blended reference states $\tilde{\mathbf{s}}$, we in effect have blended the reference control as well. With the blended reference states and actions, the character can transition more easily and the simulated motions become smoother.

6 Planning

We assume that a freerunner can only see and plan for the next impending obstacle, rather than for all the obstacles ahead. We thus focus on local planning for the first impending obstacle, and use a step-based kinematic planner with limited-horizon. Figure 6 shows the relationship between the step length and the speed and turning angle of our parameterized running controllers. We will hereafter call this relationship the step-length model, and note that step length is not affected by turning. All our motions need to approach the obstacles in a perpendicular fashion in order to carry out the skills with maximal success. We thus use a simple direction tracking algorithm to produce perpendicular approaches.

The step-length model also shows that the step-length l is nearly linear with respect to the running speed v, and the fitted model is $l = 0.2v + 0.17$. We formulate speed planning as a constrained optimization:

$$\begin{aligned} &\min w_u(u - u_h)^2 + \sum w_i(v_i - v_{i-1})^2 \\ &\text{s.t.} \quad v_n = v_{in} \end{aligned} \tag{15}$$

where n is the estimated number of steps. u is the distance to the obstacle when the clearing maneuver is ready to execute, and u_h is the optimized distance to the obstacle during the parameterization described in Section 4.2. v_n is the end velocity after n running steps, and we wish it to be as close as possible to the optimal starting velocity v_{in} for clearing skills, which is taken from $\boldsymbol{\theta}_{in}$, one of the optimized parameters of the composition described in Section 5.1. We estimate the number of steps n based on the current distance to u_h, the current velocity and the desired velocity v_{in}.

We solve the above planning model analytically at the beginning of each step for the desired running velocities. The output of the planner is a series of desired v_i that can lead the character to the right spot with the right speed ready for obstacle clearing. Since our character is physics-based and therefore has transient effects, the actual motion will not exactly follow the planned motion. We thus replan at every step according to the current character state and the goal state.

7 Results

We have implemented all the optimization and control components in C++. We use the Open Dynamics Engine (ODE) version 0.11 to simulate our character. The simulation time step is 0.5ms and the coefficient of friction is 0.8. The kinematic and dynamic properties of our character model are directly taken from Liu et al. [2010]. The simulation runs faster than real-time on a common laptop. This includes the online planning component, the application of the learned reduced-order feedback policies, and the forward dynamics. The compute costs are dominated by the forward dynamics computations.

All the offline components are tested on a 24-core machine with Intel Xeon X5660@2.80GHz CPU. Adding a new skill to the parkour system requires between 3∼8 hours of computation, depending on the difficulty, dimensionality, and desired parameter space of the motion skill. Table 1 shows a more detailed breakdown of the computational cost for each optimization component. There are multiple possible choices for the reduced-order feedback matrices in Equation 1, and here we only show results using 3rd order reduced feedback matrices. That is, $\mathbf{M_{ap}}$ is $m \times 3$ and $\mathbf{M_{sp}}$ is $3 \times n$ for all the skills. For more analysis of the impact of the chosen order for the reduced feedbacks, we refer interested readers to [Ding et al. 2012].

A practical simplification in the development of the controls for the **transOut** phases is to share the same controller across all the obstacle clearance skills. Furthermore, the SIMBICON-style controller is quite robust and we can share the controllers optimized for chosen speeds of (2.0, 2.5, 3.0, 3.4, 3.8) m/s. The total cost of this optimization is 1.9 hours, and we do not need to redo it for new skills. Only the optimization for the first two phases of the composition is task dependent.

Figure 7 shows two examples from the vaulting parameterization. The upper row is a vaulting over an obstacle of 70cm, and the lower row an obstacle of 90cm. Note that our character model is 170cm in height. The middle column of Table 1 gives the ranges of obstacle heights that our character can jump onto, vault over, and drop-roll over. We illustrate the results of composition in Figure 8. We also encourage readers to better evaluate the quality of these results from the accompanying video. The video additionally illustrates the failure mode for attempting to vault an obstacle that is too high.

Due to the dynamic nature of the motions, the simulation is not guaranteed to accomplish an arbitrary terrain crossing task with a 100% success rate. To estimate the robustness of our composed controllers, for each maneuver we did a simple test where we put the obstacle of the default height at a fixed location, and start the parkour avatar at a distance d from the obstacle, with $d \in [30, 100]$m, sampled every 0.1m. The success rate for the jump, speed-vault, and drop-roll are 87%, 90%, and 83%, respectively.

Skill	Feedback Control (minutes)	#Dim	Parameterization (hours)	#Dim	Explored Parameter Range	Composition (hours)		#Dimension	
						transIn+clear	**transOut**	**transIn+clear**	**transOut**
run	12.0	63	10.6	83	[-6, 6]° × [2.0, 5.0] m/s	2.3		136	
jump	<1.0	72	0.5	13	[0.1, 0.7] m	6.3	1.9	235	12
vault	<1.0	171	1.6	40	[0.6, 1.0] m	5.6		253	
drop-roll	<1.0	189	0.9	41	[0.9, 2.0] m				

Table 1: *Performance statistics and dimensionality of each component optimization for the parkour skills.*

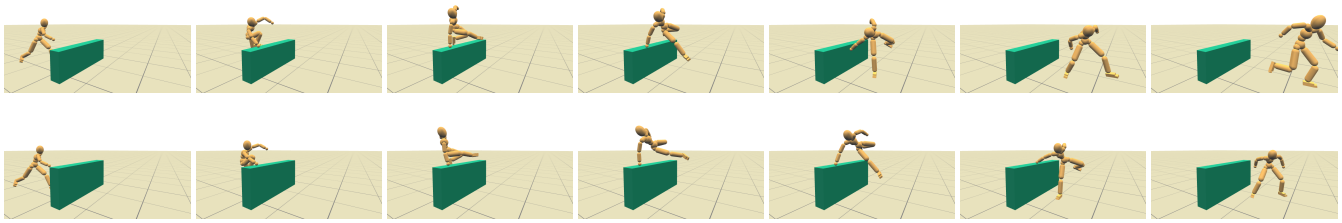

Figure 7: *Parameterization of the vault. Upper row: 70cm obstacle; lower row: 90cm obstacle. Every sixth frame is shown from a 30-fps sequence with a fixed camera.*

8 Conclusions

Creating controllers for highly dynamic human motions is challenging. We have presented a framework for learning robust control strategies for a set of highly agile, parameterized, parkour-style motions. Key to achieving the results is a strategy for learning feedback policies that are parameterized across a range of motions, an efficient multidimensional continuation procedure, and a process for learning feedback-based transition policies. These components form a flexible pipeline for developing robust and agile skills based on a single motion capture exemplar per skill.

In the future, we wish to experiment with more parkour skills, such as flips and wall climbs. We would also like to address several limitations of the methods proposed in this paper. Currently the framework is only partly automated and tackles a limited set of skills. The addition of a new skill requires authoring of the objective function, insight into the phases of a motion, and insight into relevant features and action variables for the motion. However, the shared structure found across the four parkour motions demonstrated to date leads us to believe that in many cases new motions will require only fine tuning of existing choices rather than requiring completely new objective functions and relevant state variables. We further intend to develop motion planning strategies that allow for earlier anticipation, viewing obstacles in the world in terms of its affordances, and predictions of the probability of success in advance of motion execution. We expect that the robustness of individual skills can still be further improved.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful feedback. This work was partially supported by Singapore Ministry of Education Academic Research Fund Tier 2 (MOE2011-T2-2-152) and Tier 1 (R-252-000-429-133). Michiel van de Panne was supported by NSERC and GRAND.

References

ARIKAN, O., AND FORSYTH, D. A. 2003. Interactive motion generation from examples. *ACM Transactions on Graphics 21*, 3, 483–490.

CARR, J. C., BEATSON, R. K., CHERRIE, J. B., MITCHELL, T. J., FRIGHT, W. R., MCCALLUM, B. C., AND EVANS, T. R. 2001. Reconstruction and representation of 3d objects with radial basis functions. In *SIGGRAPH '01*, 67–76.

CHAI, J., AND HODGINS, J. K. 2007. Constraint-based motion optimization using a statistical dynamic model. *ACM Transactions on Graphics 26*, 3, Article 8.

CHOI, M. G., KIM, M., HYUN, K., AND LEE, J. 2011. Deformable motion: Squeezing into cluttered environments. *Computer Graphics Forum (EUROGRAPHICS 2011) 30*, 2, 445–453.

COROS, S., BEAUDOIN, P., AND VAN DE PANNE, M. 2009. Robust task-based control policies for physics-based characters. *ACM Transctions on Graphics 28*, 5, Article 170.

COROS, S., BEAUDOIN, P., AND VAN DE PANNE, M. 2010. Generalized biped walking control. *ACM Transctions on Graphics 29*, 4, Article 130.

COROS, S., KARPATHY, A., JONES, B., REVERET, L., AND VAN DE PANNE, M. 2011. Locomotion skills for simulated quadrupeds. *ACM Transactions on Graphics 30*, 4, Article 59.

DA SILVA, M., ABE, Y., AND POPOVIĆ, J. 2008. Interactive simulation of stylized human locomotion. *ACM Trans. Graph. 27*, 3, Article 82.

DA SILVA, M., DURAND, F., AND POPOVIĆ, J. 2009. Linear bellman combination for control of character animation. *ACM Trans. Graph. 28*, 3, Article 82.

DING, K., LIU, L., VAN DE PANNE, M., AND YIN, K. 2012. Learning reduced-order feedback policies for motion skills. Tech. Rep. TR-2012-06, University of British Columbia.

EDWARDES, D. 2009. *The Parkour and Freerunning Handbook*, first ed. HarperCollins Publishers.

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

GUTMANN, A. K., JACOBI, B., BUTCHER, M. T., AND BERTRAM, J. E. A. 2006. Constrained optimization in human running. *The Journal of Experimental Biology 209*, 622–632.

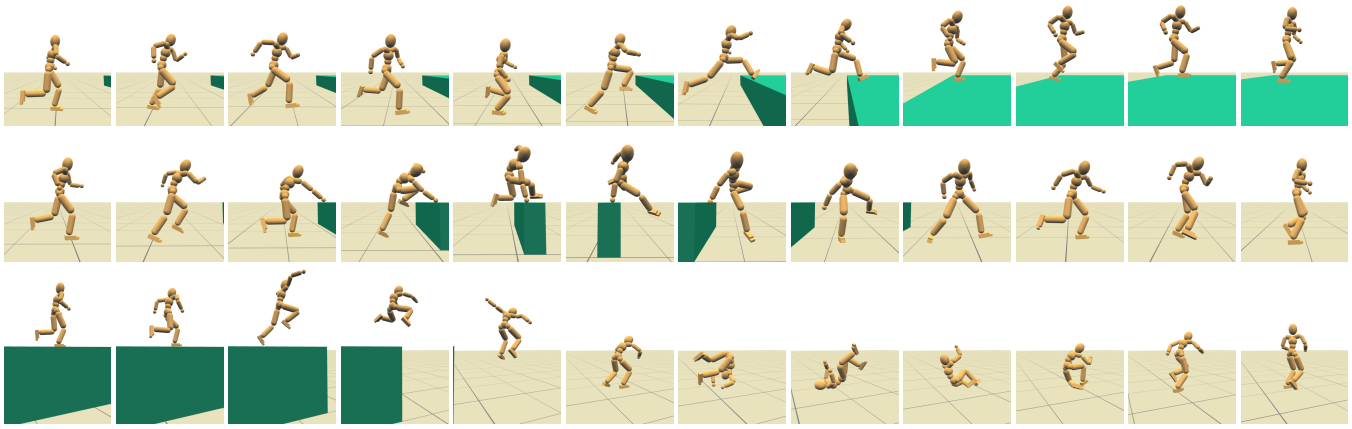

Figure 8: *Control composition. Top row: running and jumping; middle row: running and vaulting; bottom row: drop-rolling. Every third(jump), fifth(vault), and seventh(drop-roll) frame is shown from a 30-fps animation with a tracking camera.*

HANSEN, N. 2006. The cma evolution strategy: A comparing review. In *Towards a New Evolutionary Computation*, 75–102.

HECK, R., AND GLEICHER, M. 2007. Parametric motion graphs. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, 129–136.

HODGINS, J. K., WOOTEN, W. L., BROGAN, D. C., AND O'BRIEN, J. F. 1995. Animating human athletics. In *Proceedings of SIGGRAPH 1995*, 71–78.

KOVAR, L., AND GLEICHER, M. 2004. Automated extraction and parameterization of motions in large data sets. *ACM Transactions on Graphics (TOG) 23*, 3, 559–568.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. In *SIGGRAPH 2002 Conference Proceedings*, 473–482.

KWON, T., AND HODGINS, J. 2010. Control systems for human running using an inverted pendulum model and a reference motion capture sequence. In *SCA '10: Proceedings of the 2010 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 129–138.

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

LEE, Y., LEE, S. J., AND POPOVIĆ, Z. 2009. Compact character controllers. *ACM Transctions on Graphics 28*, 5, Article 169.

LEE, Y., WAMPLER, K., BERNSTEIN, G., POPOVIĆ, J., AND POPOVIĆ, Z. 2010. Motion fields for interactive character locomotion. *ACM Transctions on Graphics 29*, 6, Article 138.

LEE, Y., KIM, S., AND LEE, J. 2010. Data-driven biped control. *ACM Transctions on Graphics 29*, 4, Article 129.

LIU, L., YIN, K., VAN DE PANNE, M., SHAO, T., AND XU, W. 2010. Sampling-based contact-rich motion control. *ACM Transactions on Graphics 29*, 4, Article 128.

MCCANN, J., AND POLLARD, N. 2007. Responsive characters from motion fragments. *ACM Transactions on Graphics 26*, 3, Article 6.

MIN, J., CHEN, Y., AND CHAI, J. 2009. Interactive generation of human animation with deformable motion models. *ACM Transactions on Graphics 29*, 1, Article 9.

MORDATCH, I., DE LASA, M., AND HERTZMANN, A. 2010. Robust physics-based locomotion using low-dimensional planning. *ACM Trans. Graph. 29*, 4, Article 71.

MUICO, U., LEE, Y., POPOVIĆ, J., AND POPOVIĆ, Z. 2009. Contact-aware nonlinear control of dynamic characters. *ACM Trans. Graph. 28*, 3, Article 81.

MUICO, U., POPOVIĆ, J., AND POPOVIĆ, Z. 2011. Composite control of physically simulated characters. *ACM Trans. Graph. 30*, 3, Article 16.

SAFONOVA, A., HODGINS, J. K., AND POLLARD, N. S. 2004. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics (TOG) 23*, 3, 514–521.

SOK, K. W., KIM, M., AND LEE, J. 2007. Simulating biped behaviors from human motion data. *ACM Trans. Graph. 26*, 3, Article 107.

TREUILLE, A., LEE, Y., AND POPOVIĆ, Z. 2007. Near-optimal character animation with continuous control. *ACM Transactions on Graphics (TOG) 26*, 3, Article 7.

WANG, J. M., FLEET, D. J., AND HERTZMANN, A. 2009. Optimizing walking controllers. *ACM Transctions on Graphics 28*, 5, Article 168.

WEI, X., MIN, J., AND CHAI, J. 2011. Physically valid statistical models for human motion generation. *ACM Transctions on Graphics 30*, 3, Article 19.

YE, Y., AND LIU, C. K. 2010. Optimal feedback control for character animation using an abstract model. *ACM Trans. Graph. 29*, 4, Article 74.

YIN, K., LOKEN, K., AND VAN DE PANNE, M. 2007. Simbicon: Simple biped locomotion control. *ACM Transctions on Graphics 26*, 3, Article 105.

YIN, K., COROS, S., BEAUDOIN, P., AND VAN DE PANNE, M. 2008. Continuation methods for adapting simulated skills. *ACM Transctions on Graphics 27*, 3, Article 81.

ZHAO, P., AND VAN DE PANNE, M. 2005. User interfaces for interactive control of physics-based 3D characters. In *ACM SIGGRAPH 2005 Symposium on Interactive 3D Graphics and Games*, 87–94.