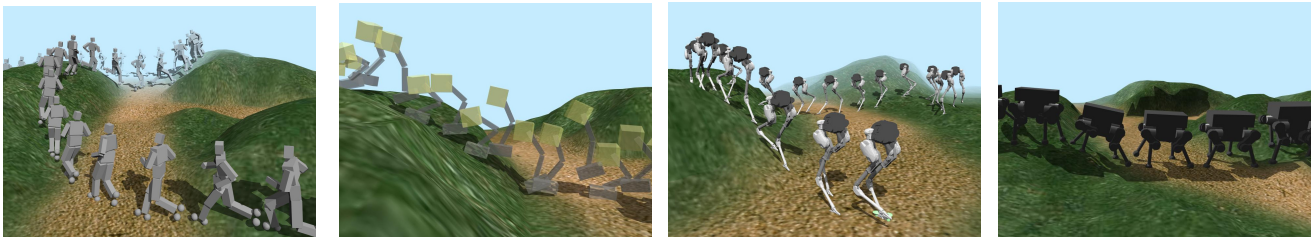
Fast and Flexible Multilegged Locomotion Using Learned Centroidal Dynamics

TAESOO KWON, Hanyang University
YOONSANG LEE, Hanyang University
MICHIEL VAN DE PANNE, University of British Columbia



(a) Humanoid character (biped) (b) Luxo character (monopod) (c) Cassie robot (biped) (d) ANYmal robot (quadruped)

Fig. 1. Our system generates rich variations of walking, running, and jumping for a variety of characters at interactive rates.

We present a flexible and efficient approach for generating multilegged locomotion. Our model-predictive control (MPC) system efficiently generates terrain-adaptive motions, as computed using a three-level planning approach. This leverages two commonly-used simplified dynamics models, an inverted pendulum on a cart model (IPC) and a centroidal dynamics model (CDM). Taken together, these ensure efficient computation and physical fidelity of the resulting motion. The final full-body motion is generated using a novel momentum-mapped inverse kinematics solver and is responsive to external pushes by using CDM forward dynamics. For additional efficiency and robustness, we then learn a predictive model that then replaces two of the intermediate steps. We demonstrate the rich capabilities of the method by applying it to monopeds, bipeds, and quadrupeds, and showing that it can generate a very broad range of motions at interactive rates, including banked variable-terrain walking and running, hurdles, jumps, leaps, stepping stones, monkey bars, implicit quadruped gait transitions, moon gravity, push-responses, and more.

CCS Concepts: • **Computing methodologies → Physical simulation**; *Motion path planning*.

Additional Key Words and Phrases: character animation, motion planning, rigid-body simulation

Authors' addresses: Taesoo Kwon, Department of Computer Science and Engineering, Hanyang University, taesoo@hanyang.ac.kr; Yoonsang Lee, Department of Computer Science and Engineering, Hanyang University, yslee111@gmail.com; Michiel van de Panne, Department of Computer Science, University of British Columbia, van@cs.ubc.ca.

0730-0301/2020/7-ART46 $15.00
https://doi.org/10.1145/3386569.3392432

ACM Reference Format:
Taesoo Kwon, Yoonsang Lee, and Michiel van de Panne. 2020. Fast and Flexible Multilegged Locomotion Using Learned Centroidal Dynamics. *ACM Trans. Graph.* 39, 4, Article 46 (July 2020), 24 pages. https://doi.org/10.1145/3386569.3392432

1 INTRODUCTION

Physically simulated characters can respond to user input while also adapting to environmental and external forces. However, it is not easy to create a controller which robustly generates natural locomotion because legged locomotion is inherently underactuated. It means the character cannot be directly controlled, but instead must be controlled through contact forces applied to the feet or hands. This indirect control is challenging because the contact forces can only push, the contact point should be within the reach of the character, and all contacts must occur at appropriate time intervals. Controlling locomotion skills for wide variations of terrains, turns, speeds, and accelerations is even more challenging.

Trajectory optimization is one approach to address this problem, and formulates it using an objective function and a set of constraints to define the desired motion. Model predictive control (MPC) is an online trajectory optimization approach, which is powerful for realizing motions that require significant anticipation and planning. In this approach, an optimized motion is computed for the near-term future at each planning time-step. However, trajectory optimization remains a challenging computational problem because this formulation inherently requires a high-dimensional search space that is related to the degrees of freedoms of a character, and a number of variables that increases linearly with the time horizon, and the complexity of the environment.

To mitigate the computational cost, trajectory optimization is often combined with simplified dynamic models such as a spring-loaded inverted pendulum model (SLIP) [Mordatch et al. 2010], an inverted pendulum on a cart model (IPC) [Kwon and Hodgins 2010],

and a centroidal dynamics model (CDM) [Orin et al. 2013] as these models can largely reduce the dimension of the search space. In particular, the CDM has recently been a popular simplification in the robotics community for humanoid and quadruped control. It projects the full-body dynamics of an articulated model to its center of mass (COM) and thus the model and its motion can be simplified to the centroidal inertia and the overall linear and angular momentum. The CDM can be further simplified to the dynamics of a single rigid body by using a frame anchored at the COM to represent overall body orientation [Winkler et al. 2018]. We adapt the IPC and CDM as building blocks for our MPC-based control method.

In this paper, we propose an online control system that can generate rich variations of walking, running, and jumping for a variety of characters at interactive rates. The use of a three-level planner with simplified dynamics models efficiently realizes the physical fidelity of the resulting motion. Our system plans the CDM trajectory (COM positions and orientations), contact timings, and contact positions of the character using an iterative MPC-style sliding horizon using the simplified dynamics models, and then recovers the full-body motion from the planned information. The first step provides an approximate plan of the COM and center of pressure (COP) trajectories using an IPC. The footstep planner then plans the footstep locations and timings based on the COP trajectory. These plans are fed into the CDM trajectory planner, and it further optimizes the planned COM trajectory and footstep locations while simultaneously finding the contact forces that realizes them. Note that the CDM trajectory planner guarantees the physical correctness of the CDM plan. Finally, our system generates the full-body motion that corresponds to the final CDM plan using a new momentum-mapped inverse kinematics solver.

Our approach has an important difference with deep reinforcement learning (DRL) methods, which have recently attracted much attention. Because it is based on online optimization, our method allows users to freely experiment with different characters and terrain scenarios to generate a wide range of physically-valid locomotion skills at interactive rates, without preprocessing.

Solving the CDM optimization can take too much time for real-time applications, and can occasionally fail to find feasible solutions in some cases due to the complexity of the nonlinear programming (NLP) problem. To tackle this issue, we propose to use a deep neural network that learns the centroidal dynamics of a character in a preprocessing step, as trained on a number of randomly-generated CDM plans and their corresponding synthesized full-body motions. This predictive model is then queried during online use, allowing our system to produce the full-body motion much more efficiently and with guaranteed robustness.

Our system can generate motions without a reference motion, i.e., given only the character model and its rest pose. When desired, it can also use short motion clips (1–2 seconds long) to improve the style of the generated motion. Unlike much prior work, our characters can respond to user input quickly enough to perform sharp turns or abrupt accelerations within one or two steps, without using any reference motions of similar behavior or motion-specific optimizations. These advantages naturally arise from the appropriate combination of the simplified dynamics models.

Our contributions are as follows:

- The generation of natural and dynamic multilegged locomotion, at interactive speeds, for a wide spectrum of scenarios and characters
- Three-level planning using the combination of two commonly-used simplified dynamics models, an IPC and a CDM, together with MPC-style replanning. This allows for responsive and adaptive motion synthesis due to efficient computation.
- A learned predictive model for further interactive performance and improved solution robustness, which takes future motion information for variable-length durations and predicts a full-body motion segment for the planning horizon.

2 RELATED WORK

Animated character motion can be synthesized in many ways. In what follows, we review only the closest work using dynamics-based methods, i.e., those that include some component of physics-based simulation, and focusing on locomotion. Because these methods model the known underlying physics, they have the potential to generalize better and require less data than kinematic methods. We point the reader to surveys in animation [Geijtenbeek and Pronost 2012] and legged robotics [Wieber et al. 2016] for general overviews of physics-based motion generation for articulated figures.

We use a three-way categorization in our description of previous work. Motion control requires anticipation of future. In *feedback-based control policies*, anticipation is implicit in the final control policy, as they only compute the current action to take, given the current state. In *trajectory-optimization* approaches, the anticipation is explicit, as achieved via an optimization over a future time window of the sequence of actions to be taken and the resulting states over time. We include model-predictive control (MPC) methods in this category because of their use of trajectory (re)optimization at regular time intervals to yield a robust control policy. Lastly, several methods use a *combination* of these approaches.

2.1 Feedback-based Control

A broad range of work tackles the problem of developing feedback-based control policies for agile legged locomotion. As such, we focus on representative examples of work in this area. Feedback-based methods for locomotion still need to anticipate well into the future, which can be accomplished in many ways. Much can be done with using finite state machines as a coarse model of what is expected to unfold over time. When combined with local feedback laws designed or learned for each state, this has proven to be a remarkable effective approach [Hodgins et al. 1995; Yin et al. 2007]. The feedback computations for individual states are often designed around the use of a simplified dynamics model, such as an inverted pendulum, which allows the control policy to anticipate further into the future, e.g., [Coros et al. 2010, 2011; Ha et al. 2012; Kajita et al. 2003; Mordatch et al. 2010; Tsai et al. 2010]. Available motion capture data can also be integrated into such approaches, e.g., [Coros et al. 2011; Lee et al. 2010; Tsai et al. 2010]. In many cases, the motion capture data itself can also serves as a "sketch" of how the motion is expected to unfold.

Many control policies also directly take the full-body dynamics into account, most often by solving for the actions at the current time step using a quadratic program (QP). This can take into account the physics, contact force constraints, a sketch of the future desired motion of the linear and angular momementum, and other terms and constraints, e.g., [Abe et al. 2007; de Lasa et al. 2010; Jain et al. 2009; Macchietto et al. 2009; Muico et al. 2009].

More recently, significant progress has been towards achieving general solutions using deep reinforcement learning (DRL). For these models, the implicit anticipation is learned offline over time via experience, and also characterized explicitly via a state or state-action value function, e.g., see [Heess et al. 2017], among many works. In order to be useful for graphics and animation, more realistic motions are required. This can be accomplished in a variety of ways, e.g., including knowledge into the control parameterization [Peng et al. 2016], encouraging low-energy, symmetric motions [Yu et al. 2018], directly incorporating motion capture data into the objective, e.g., [Peng et al. 2018, 2017], exploiting motion data to learn a diversity of motions [Park et al. 2019]. DRL methods have also been successfully deployed on real robots, e.g., [Tan et al. 2018; Xie et al. 2019].

2.2 Trajectory Optimization

Trajectory optimization has an equally long history as a method for synthesizing physically-valid movements. While the optimization problem is challenging, mainly due to the number of variables and their non-linear (or even discontinuous) evolution over time, the past decade has seen continual advances in the capabilities of such methods. Improvements have exploited (a) simplified dynamics models; (b) example motion data; (c) ever-improving constrained-optimization solvers; and (d) improved objective and constraint formulations. Our own work is no exception and it will exploit all of these. To aid the reader in comparing our work to previous results, we include Table 5 (see Appendix), which provides links to the animated results of many of the most-closely related works to our own. Also, while feedback-based control policies often require extensive offline learning, trajectory optimization methods are directly model-based, meaning that changes made to a model, e.g., the mass of a link or the length of a leg, can instantly be reflected in solutions that can be computed online.

Centroidal dynamics models (CDMs) allow for a significant simplification of the trajectory optimization problem. They model the motion in terms of the center of mass (COM), its overall linear momentum, as well as the overall orientation of the body and its angular momentum. The evolution of the corresponding 12 state variables (6 DOFs and their velocities) over time, taken together with knowledge of the foot contact locations and forces, model the key aspects of most dynamic legged motions. The utility of CDMs is well understood for humanoid motions [Orin et al. 2013]. High quality inverse kinematics solutions can then be used to produce the full-body motions. These can (optionally) be tracked using inverse dynamics or other tracking controllers to produce physically-validated full-body motions. Early work using centroidal dynamics demonstrates its flexibility in creating adaptable versions of specific motions, including a variety of gymnastic motions with keyframe constraints [Liu and Popović 2002], human walking [Ye and Liu 2010] and running [Kwon and Hodgins 2010] using mocap reference motions, performance animation [Ishigaki et al. 2009], and jumping [Mordatch et al. 2010] without using reference motions. Similar to our method, these approaches also use simplified physical models such as an inverted pendulum [Kwon and Hodgins 2010] or a spring-loaded inverted pendulum (SLIP) [Mordatch et al. 2010]. However, the physical models used in these papers have a lower complexity than the CDM we use. For example, the SLIP model [Mordatch et al. 2010] is based on a lumped mass assumption, so there is no consideration for angular momentum in its dynamics. Other physical models such as the divergent component of motion assume that the changes in angular momentum are zero [Englsberger et al. 2015; Takenaka et al. 2009].

Significant progress has been achieved without a priori gait specification by defining contact-invariant optimization strategies [Mordatch et al. 2012]. The resulting ungaited motions are demonstrated on a wide variety of motions for bipeds and quadrupeds, performing locomotion and manipulation tasks in a variety of challenging settings. The motions take 2-10 minutes to optimize per clip and benefit from a crafted continuation schedule. With the exception of a kick-up for a handstand, the motions demonstrate limited use of momentum, e.g., they have a quasistatic nature.

More recently, a single-solve is proposed for both feasible centroidal dynamics and joint angles, and incorporating some capability for collision constraints [Dai et al. 2014]. This implements a potentially very general solution and is demonstrated on a simulated Atlas robot, as well as straight-line motion for a Little Dog robot. Motions take several minutes to compute. An ability to work with an unspecified contact sequence is shown for one example. Other recent work demonstrates an ability to solve for simultaneous contacts and motions for the HyQ quadruped robot [Aceituno-Cabezas et al. 2018], yielding forward-moving walking solutions in less than a second. This is accomplished using mixed-integer convex programming. The convex decomposition of the angular dynamics is acknowledged to work well for motions having low angular momentum [Ponton et al. 2016], but other methods are needed for more aggressive motions. Real-time MPC methods [Farshidian et al. 2017] are demonstrated on the HyQ quadruped robot, generating optimized trajectories for the next few phases of the motion within only a few milliseconds. This uses a sequential linear quadratic method, leveraging linearized system dynamics and constraints and a quadratic approximation of the cost function. A forward walk and robust in-place stepping are demonstrated.

Trajectory optimization of centroidal dynamics models is recently demonstrated for monopeds, bipeds, and quadrupeds, including a demonstration of transfer to the ANYmal robot [Winkler et al. 2018]. This is closest in nature to our own work in many ways. A single trajectory optimization resolves the gait sequence, foothold timings and locations, and 6D body motion. The interior-point IPOPT solver yields fast optimization times; motions with time horizons of around 5 s are solved in approximately 20 s. The method is demonstrated on a variety of motions, including terrain adaptation for monopeds, bipeds, and quadrupeds. Our work differs in terms of the various contributions listed earlier, including interactivity, motion quality, and demonstrated generality.

Our work has connections with the momentum-mapped inverted pendulum model [Kwon and Hodgins 2017], which employs an optimized inverted pendulum model to design control systems around reference motion capture clips. This supports the creation of full-body QP-based tracking control systems for motions that include jumping and gymnastics. Computing the controllers requires hours of preprocessing time. They are also (to the best of our knowledge) insufficiently robust to adapt to the broad range of motion variations that we illustrate. Lastly, they require a specific motion capture example for each new type of motion.

Full-body dynamics trajectory optimization or model-predictive control has also been explored in depth using a variety of approaches. Many of the methods are offline, and are optimized with respect to a variety of objectives, e.g., [Al Borno et al. 2013; Borno et al. 2017; Felis and Mombaur 2016; Liu et al. 2015, 2010]. Relatedly, significant effort has also been invested into developing online full-body dynamics solutions, both in simulation [Hämäläinen et al. 2014, 2015; Tassa et al. 2012] and deployed on robots [Carpentier et al. 2016; Erez et al. 2013; Koenemann et al. 2015; Serra et al. 2016]. Multiple levels of planning are also found in several approaches, e.g. [Herzog et al. 2016; Tonneau et al. 2018; Zimmermann et al. 2015]. All online methods need to efficiently solve a remarkably difficult problem with relatively limited computation, and thus they typically may employ a short time horizon, may produce motions limited in flexibility, and may need to be given contact points or a contact FSM, or a sequence of intermediate goals. It is possible to further improve the motion quality using reference motions and smoothed contact dynamics, as demonstrated in [Han et al. 2016; Hong et al. 2019], or used in combination with offline learning [Lowrey et al. 2018].

2.3 Combined Approaches

Trajectory optimization methods can be used to produce data for learning feedback-based controllers, typically as part of an iterative process, e.g., [Levine and Koltun 2013; Liu and Hodgins 2018; Liu et al. 2016, 2012; Mordatch et al. 2015; Zhang et al. 2016]. Learning can also be used in support of trajectory optimization, either to inform additional examples, e.g., [Rajamäki and Hämäläinen 2017], or to develop value functions for the terminal states of the trajectory optimization [Lowrey et al. 2018]. These combined methods provide an interesting avenue for further research and we expect that they can be further enhanced with the methods that we propose.

3 SYSTEM OVERVIEW

Our proposed system generates motions by planning footstep locations, timing, contact forces, and other necessary elements, using a multi-level trajectory optimization scheme. In what follows, we first give an overview of our CDM-based motion generation system. We then give a high-level description of the supervised-learning framework that serves to further improve the performance of the generation system.

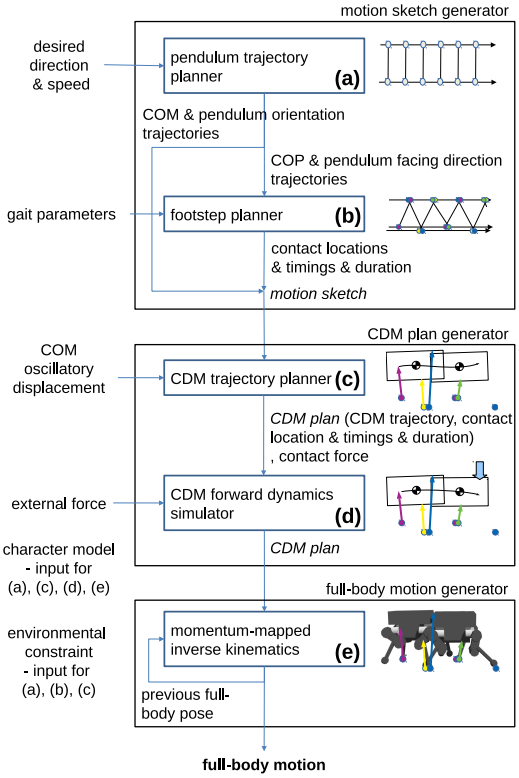


Fig. 2. Our system consists of multi-level generators. The motion sketch generator is based on an inverted pendulum on a cart model (IPC) to produce the "rough" sketch of the character motion, and CDM plan generator uses a CDM to generate more detailed behaviors. The full-body motion generator produces the final full-body motion of the character.

3.1 Generation System

The input of our system includes the character model, control parameters such as walking speed and direction, gait parameters such as step duration, environmental constraints, external force information, and optionally a reference motion. As shown in Figure 2, our system is composed of motion sketch generator, CDM plan generator, and full-body motion generator. The motion sketch generator uses a hierarchy of two-level planners: pendulum trajectory planner and footstep planner. The CDM plan generator uses the CDM planner. All three planners predict trajectories in a short future horizon at every locomotion cycle, except for the biped characters in which case planning takes place at every footstep for improved responsiveness.

The motion sketch generator (Section 4) calculates a rough "sketch" of the upcoming COM trajectory (COM position) and footstep locations, timings, and duration, as well as the pendulum orientation trajectory. Taken together, we call this information the *motion sketch*, which is used as input to the CDM plan generator.

Its first planning step, the pendulum trajectory planner (Section 4.1), is based on an inverted pendulum on a cart model (IPC) and plans the balancing behaviors of characters using the motion of the pendulum and the cart (step (a) in Figure 2, Figure 4). This

step takes the character model, desired direction and speed, and environmental constraints as input to compute the COM, pendulum orientation, and center of pressure (COP) trajectories. These trajectories are computed over a fixed-duration time-horizon which is long enough to contain 1–2 footsteps per limb for all limbs.

The next step is the footstep planner (Section 4.2), which plans the footstep locations and timings, and their contact duration based on the COP trajectory from the previous step (step (b) in Figure 2). Other inputs to this step include: environmental constraints such as terrain gaps, the pendulum facing direction from the previous step, and gait parameters such as the desired stride length and duration. The footstep locations are obtained by sampling the planned COP trajectory using an optimization process. The number of footsteps to solve for corresponds to a time-horizon of 1–2 footsteps per limb for all limbs. The order of the pendulum trajectory planner and the footstep planner can be reversed, or these two planners even can be omitted for some examples.

The CDM plan generator (Section 5) computes the CDM trajectory (COM position and orientation) and refined footstep location based on a CDM. We call the output of this generator *CDM plan*, which also contains the contact timings and duration from the motion sketch. The planned CDM trajectory can be modified in a physically correct manner afterwards to reflect the effect of unexpected external force. The output CDM plan is fed into the full-body motion generator to generate final full-body motion.

The CDM trajectory planner (Section 5.1) optimizes the CDM trajectory and contact forces that cause the CDM of the character to move as closely as possible to the motion sketch from the previous step (step (c) in Figure 2). It additionally takes the character model, environmental constraints, and the horizontal oscillatory displacement of the COM. The locations of the foot contacts can also be modified to suit the law of physics and terrain constraints. The generated CDM trajectory is guaranteed to be physically correct. This optimization is performed over a time-horizon of 1–1.2 locomotion cycles.

To make a character that can respond to external forces, the CDM forward dynamics simulation (Section 5.2) can modify the CDM plan reflecting the external forces (step (d) in Figure 2). The CDM trajectory is modified in a physically correct manner under the contact force calculated in the previous step and the external force specified by the user.

The full-body motion generator (Section 6) mainly uses a single component, the momentum-based inverse kinematics solver (Section 6.1). The solver generates the final full-body pose that matches the CDM plan at the current frame (step (e) in Figure 2). In addition to the CDM plan, it takes the full-body pose solved at the previous frame in order to consider joint velocities as well as joint angles. An optional reference motion can be used to guide the stylistic details of the final motion. By default, the rest pose of the character model is used. This step is performed at every rendering frame, unlike the aforementioned planners which are performed at every locomotion cycle or step.

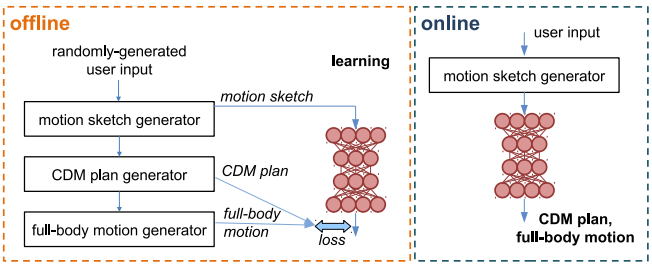


Fig. 3. Our learning framework learns the output CDM plan and corresponding full-body pose for each motion sketch offline so that it can generate the final full-body motion efficiently online.

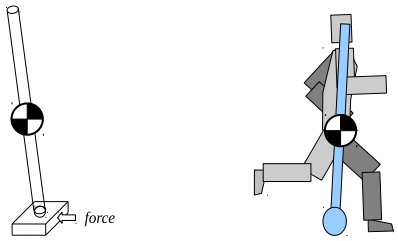


Fig. 4. The structure of an IPC (left) and an example IPC constructed from a full-body pose (right).

3.2 Learning Framework

Our system can generate final full-body motions at interactive rates without any preprocessing. However, the CDM trajectory planning can take too much time for smooth real-time performance, and may sometimes fail to find the feasible solution due to the complexity of the nonlinear programming. To effectively cope with this issue, we train a deep neural network to directly generate the full-body motion (Figure 3). This allows us to avoid having to compute the CDM optimization problem online. Here, we briefly summarize the learning framework, and its details are described in Section 7.

In the offline learning stage, a training dataset is created for the learning framework using the generation steps as already described. A deep neural network is then trained to predict the CDM plan and the corresponding full-body poses.

In online use, once a user specifies control parameters, the motion sketch generator produces the motion sketch for the user parameters. The final full-body motion can then be generated immediately from the motion sketch.

4 MOTION SKETCH GENERATION

The motion sketch represents the desired position of the COM and orientation of the pendulum over a time-horizon, as represented by an inverted pendulum on a cart model (IPC), as well as the contact positions and contact timings for all character model end-effectors.

4.1 Pendulum Trajectory Planner

We use an IPC to roughly model the balancing behaviors of various characters during stepping or standing motions, as in [Kwon and

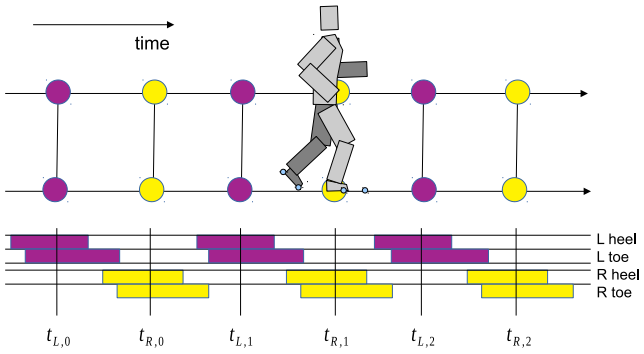


Fig. 5. A schematic diagram of sample timings and corresponding cart positions of the pendulum. Upper: The pendulum COM positions (upper circles) and the cart positions (lower circles) at sample timings. Purple circles represent left foot contacts, and yellow right. Lower: The contact timings for the heel and toe of each limb.

Hodgins 2017] (Figure 4). A linear quadratic regulator (LQR) finds horizontal force applied to the cart to meet the desired direction and speed, thus COM ($\mathbf{p}_{\text{pend}}$), COP ($\mathbf{c}_{\text{pend}}$), and pendulum orientation ($\mathbf{R}_{\text{pend}}$) trajectories are generated as a result.

Given a character model, its IPC is constructed so that the COM of the pendulum and the position of the cart match the COM and COP of the character calculated using the rest pose or the average pose of the reference motion if supplied, respectively. It has the same mass and inertia properties as those of the character. The rotational joint for the leaning angle of the pendulum is not actuated, and the pendulum is only balanced by applying horizontal forces to the cart. Thus, the IPC expresses body lean when accelerating or changing direction. The motion of the IPC is always continuous, smooth, and independent of the contact state. See Appendix B for details of the planner, including LQR control and the generation of a facing direction.

4.2 Footstep Planner

The IPC provides the overall lean angle, as well as guidance for locating the ground contact positions, which will be located near the trajectory of the cart of the pendulum [Kwon and Hodgins 2017]. The cart position during the middle of a given contact phase is expected to be nearby the contact position of the support limb. Based on this observation, we generate a stepping behavior by sampling the trajectory of the cart (Figure 5).

Given the trajectories of the IPC, the footstep locations and timings are determined by sampling the trajectory of the cart (COP trajectory $\mathbf{c}_{\text{pend}}$). The actual footstep location is determined by transforming the sample location by the user-defined foot offset transformation (see Appendix C.1 for details). Instead of using regular sampling as in prior art [Kwon and Hodgins 2017], we use an optimization to further adapt the footstep locations and timing to be more natural. As depicted in Figure 2, this optimization process requires the gait parameters as input, which are desired stride length and duration, desired step length and duration, desired ratio between different contact phases, and foot offset transformation.

The optimization problem can be flexibly defined using several constraints while further minimizing the error represented by the objective function. The objective function mainly defines a compromise between a high-frequency gait and a long stride gait. This formulation has several advantages over rule-based or procedural approaches. First, it generalizes well to a variety of situations such as running over gaps and executing sharp turns. Also, multiple gait styles which may occur in different situations can be explained using a single objective function, which reduces the work required to model natural gaits and gait transitions. By changing the optimality criteria, various gait styles can be obtained from a single trajectory of the pendulum model. For example, an objective favoring frequent sampling of the footprints results in high-frequency gaits with a shorter stride.

We now provide a detailed view of the footstep planner.

4.2.1 Optimization Variables. The optimization variables are a set of contact times $\{t_{i,j}\}$, that define the midpoint of the contact duration for the corresponding footstep (i, j), as illustrated in Figure 5. For limbs with a heel and a toe, $t_{i,j}$ defines the midpoint of the overlapping contact intervals. Here, $i \in [1, n_{\text{limb}}]$ specifies the limb index, which is also denoted by L or R when convenient. The index j is the footstep count for a limb, as measured within the current planning horizon of the pendulum trajectory planner. This horizon spans a window $1 \leq j \leq n^i_{\text{contact}}$, where n^i_{contact} is the expected number of footsteps for the i-th limb in the planning horizon. Figure 5 illustrates a typical scenario, where $t_{L,1}$, $t_{R,1}$, $t_{L,2}$, and $t_{R,2}$ are timings to be determined in the current horizon planning, and $t_{L,0}$ and $t_{R,0}$ are timings already determined by the previous planning iteration. See Appendix C.2 for full details of the optimization variables.

4.2.2 Objective Function. The footsteps are adjusted by the user-specified gait parameters such as stride length and duration along with additional adjustments for avoiding leg crossing and changing gait frequency for accelerating or decelerating, by using the following objective terms. Here, stride refers to a single locomotion cycle.

Stride duration objective. The first objective term penalizes the error between the user-specified desired stride duration δ_{stride} and the actual stride duration:

$$E_1 = \sum_{i=1}^{n_{\text{limb}}} \sum_{j=0}^{n^i_{\text{contact}}-1} \left\| t_{i,j+1} - t_{i,j} - \delta_{\text{stride}} \right\|^2, \tag{1}$$

where i is the limb index, j is the footstep index, and $t_{i,j+1} - t_{i,j}$ is the actual stride duration of i-th limb.

Step duration objective. A desired step duration δ_{step} can be specified for each pair of footsteps belonging to two different limbs i_1 and i_2 and the error for it can be penalized as:

$$E_2 = \sum_{(i_1, i_2, k, \delta_{\text{step}}) \in P_{\text{dur}}} \sum_{j=0}^{J} \left\| t_{i_1, j+k} - t_{i_2, j} - \delta_{\text{step}} \right\|^2, \tag{2}$$

where P_{dur} is a user-specified set of footstep pairs, j denotes the footstep index, J is defined by $\min(n^{i_1}_{\text{contact}} - k, n^{i_2}_{\text{contact}})$. In each

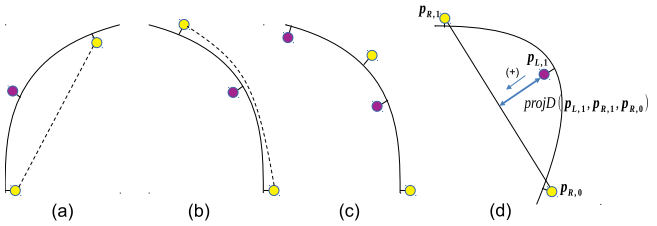


Fig. 6. Examples of footstep locations when making turn. The solid curve is the cart trajectory and the dotted curve is the expected swing path. (a) If the inner foot is in contact at the start of turning, the next outer footstep in the middle of the curve would result in an efficient, close to straight swing path of the inner foot. (b) If the turning begins with the outer footstep, however, the next inner footstep in the middle would result in an circuitous swing path of the outer foot. (c) The desirable behavior in such cases is taking a shorter inner footstep followed by a shorter outer footstep in the middle of the curve not to make such a circuitous swing path. (d) This behavior can be modeled by penalizing projD $(\cdot, \cdot, \cdot)$, which is a projected signed-distance between each footstep location and two adjacent footstep locations of the opposite limb.

element in P_{dur}, $k \geq 0$ represents the difference in the temporal order and δ_{step} is the desired duration for the limbs i_1 and i_2. For example, two half cycles of a biped locomotion as shown in Figure 5 can be represented using $P_{\text{dur}} = \{(L, R, 1, \delta^1), (R, L, 0, \delta^2\}$. A user can set the same desired step durations ($\delta^1 = \delta^2$) to create symmetrical gait or different values ($\delta^1 \neq \delta^2$) to create asymmetrical gait.

Stride length objective, Step length objective. Similarly, the third and fourth terms penalize the errors between the desired and actual stride length and the desired and actual step length:

$$E_3 = \sum_{i=1}^{n_{\text{limb}}} \sum_{j=0}^{n^i_{\text{contact}}-1} \left\| \text{dist}\left(\mathbf{p}_{i,j+1}, \mathbf{p}_{i,j}\right) - d_{\text{stride}} \right\|^2, \tag{3}$$

$$E_4 = \sum_{(i_1, i_2, k, d_{\text{step}}) \in P_{\text{len}}} \sum_{j=0}^{J} \left\| \text{dist}\left(\mathbf{p}_{i_1,j+k}, \mathbf{p}_{i_2,j}\right) - d_{\text{step}} \right\|^2, \tag{4}$$

where dist $(\cdot, \cdot)$ measures the signed distance between two footsteps along the front facing direction, d_{stride} is the desired stride length, P_{len} is a user-specified set of footstep pairs, J is defined by $\min(n^{i_1}_{\text{contact}} - k, n^{i_2}_{\text{contact}})$. In each element in P_{len}, $k \geq 0$ represents the difference in the temporal order and d_{step} is the desired step length for the limbs i_1 and i_2. For a biped walking, we use the same set of pairs for both P_{dur} and P_{len}.

Leg crossing objective. This term penalizes circuitous swing foot movement when turning to reduce the chance of leg crossing. As shown in Figure 6(a) and 6(b), the outer foot is favored as the stance foot in the sharpest turn of the curve when making sharp turns. In other words, the inner foot is preferred as the swing foot during the turn not to take an inefficient swing of a leg. This preference results in very different footstep patterns depending on whether the support foot at the start of the turn is an inner foot or an outer foot (Figure 6(a)-(c)). This behavior can be simply modeled as using a preference for the swing leg to avoid self-intersections without needing to take a circuitous path. It is modeled by penalizing the projected signed-distance between each footstep location and the line between two adjacent footstep locations of the opposite limb, to reduce the inner foot swing across the line between adjacent outer footstep locations (Figure 6(d)):

$$E_5 = \sum_{(i_1, i_2) \in P_{\text{opp}}} \sum_{j=0}^{J} \left\| \max\left(\text{projD}\left(\mathbf{p}_{i_1,j+1}, \mathbf{p}_{i_2,j+1}, \mathbf{p}_{i_2,j}\right), 0\right) \right\|^2, \tag{5}$$

where P_{opp} is an ordered set of opposite limb pairs which contains all pairs of inner limbs and outer limbs. For example, both (L, R) and (R, L) are in P_{opp} for a biped model. For a quadruped model, P_{opp} has four pairs of limbs, two for front legs and another two for rear legs. projD $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ calculates the projected signed-distance between point $\mathbf{a}$, and line segment $(\mathbf{b}, \mathbf{c})$ (Figure 6(d)). The direction of positive sign depends on which foot the middle one is, on the point $\mathbf{a}$. The positive direction is left for a left foot, and right for a right foot. Therefore, if a left foot is on the point $\mathbf{a}$ when making a right turn as shown in Figure 6(a), E_5 is evaluated to zero for these three footsteps, which means all such cases are equally preferred. J is defined by $\min(n^{i_1}_{\text{contact}} - 1, n^{i_2}_{\text{contact}} - 1)$.

Gait frequency objective. This term models the observation that the gait stride is shortened when rapidly accelerating or decelerating, in favor of using higher frequency gaits:

$$E_6 = \sum_{i=1}^{n_{\text{limb}}} \sum_{j=0}^{n^i_{\text{contact}}-1} \sum_{t \in (t_{i,j}, t_{i,j+1})} \left\| \mathbf{f}^{\text{pend}}_t \right\|^2, \tag{6}$$

where $\mathbf{f}^{\text{pend}}_t$ represents the control force at time t applied to the cart of the pendulum. To reduce E_6, the time between each footstep or the magnitude of the cart control force need to be reduced because the number of footstep samples during the current planning horizon is fixed to a sufficient number. Accelerating or decelerating a character increases the magnitude of the cart control force, which implies the time between each footstep should be decreased to keep the value of E_5 at a similar level.

Objective function. The objective function is the weighted sum of all objective terms. By adjusting the weight for each term, one can obtain a continuous range of stylistic variations, for example, to mimic a running motion of a football player who often makes sharp turns, or to obtain the style of an efficient Olympic runner.

4.2.3 Constraints. The footstep planner controls the temporal order of the footsteps and prevents the footsteps from colliding with obstacles by using the following constraints. See Appendix C.3 for more details of the following constraints.

Constraint for temporal order in a single limb. The first constraint determines the order of each footstep. When a foot contacts the ground multiple times during the time horizon, the temporal order is guaranteed not to be reversed:

$$t_{i,j+1} - t_{i,j} > 0, \ \forall i \in [1, n_{\text{limb}}], \forall j \in \left[0, n^i_{\text{contact}}\right]. \tag{7}$$

Constraint for temporal order across multiple limbs. The temporal order can be constrained across multiple limbs as well, which can be written in a general form:

$$t_{i_1,j+k} - t_{i_2,j} > 0, \ \exists i_1, i_2 \in [1, n_{\text{limb}}], \exists k \geq 0, \forall j \in [0, J], \tag{8}$$

s where k represents the difference in the temporal order and J is defined by $\min(n^{i_1}_{\text{contact}} - k, n^{i_2}_{\text{contact}})$. Any arbitrary pair of limbs can be used as i_1 and i_2 to exhibit various styles of locomotion.

Constraint for obstacle avoidance. The last constraint ensures that the footstep position does not intersect with any obstacle:

$$\mathbf{p}_{i,j} \notin B,\ \forall i \in [1, n_{\text{limb}}], \forall j \in \left[1, n^{i}_{\text{contact}}\right], \tag{9}$$

where $\mathbf{p}_{i,j}$ is the footstep position corresponding to sampling time $t_{i,j}$, and B is the set of obstacles.

4.2.4 Derivative-Free Optimization. Due to the high dimensionality and non-continuous nature of the search space and the presence of local minima, we use a stochastic derivative-free optimization algorithm, the covariance matrix adaption evolution scheme (CMAes) [Hansen and Ostermeier 1996]. CMAes uses many random samples at every iteration. We use the population size of $N_{dim}/2$, where N_{dim} is the dimensionality of the search space, and the maximum iteration limit of 100, which empirically provides good balance between quality of the solution and online performance. The derivative-free optimization supports directly implementing constraints by guaranteeing that all the random samples satisfy the constraints. See Appendix C.4 for details on how to impose the constraints.

4.2.5 Footstep contact duration. The result of the derivative-free optimization is a set of optimized sampling times $\{t_{i,j}\}_{\forall i,j}$. These optimized sampling times need to be converted to contact and swing phases for all end effectors such as toes and heels, for the CDM planner to calculate the contact force during the contact duration. We heuristically adjust the contact duration of each limb $\{d_{i,j}\}_{\forall i,j}$ based on the forward speed so that the COM moves a constant distance during the support phase. See Appendix C.5 for details.

4.2.6 Variations. The previous parts of Section 4 describe the "standard" process of the motion sketch generator used in ordinary cases. However, the objective functions and the optimization process can be easily adjusted for handling wider range of cases such as stepping stones or quadruped gaits thanks to the flexibility of our system. The implementation details for such cases are described in Appendices C.6 and G.

5 CDM PLAN GENERATION

The CDM plan consists of CDM trajectory (COM position and orientation), as represented by a CDM, as well as footstep locations, contact timings and duration. The CDM trajectory and footstep location are optimized based on the motion sketch input. For contact timings and durations, values contained in the motion sketch are used without modification.

5.1 CDM Trajectory Planner

The trajectories and footprints of the motion sketch can effectively represent the main characteristics of the character's full-body motion. However, because the laws of physics are not considered when deciding the footprints, the full-body motion might not seem physically feasible if it is generated directly from the motion sketch.

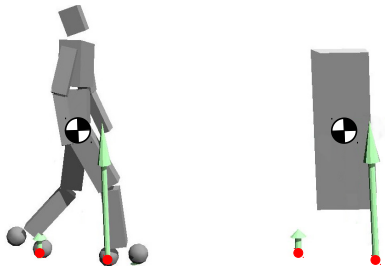


Fig. 7. A full-body character model (left) and the corresponding CDM (right). The contact forces (light green arrows) can be applied to the single rigid body at contact positions (red dots), which changes the accelaration of the body.

To give physical correctness to our simplified representation of full-body motion, we adopt a CDM. The CDM trajectory defines the sequence of COM positions and orientations, which represents the motion of the CDM frame, as anchored at its COM (Figure 7). The CDM trajectory planner computes a physically correct CDM trajectory for a finite time-horizon that closely follows the COM and pendulum orientation trajectories given by the motion sketch. We formulate solving for the CDM trajectory as a nonlinear optimization, similar to [Winkler et al. 2018], with an additional objective that makes the CDM trajectory track the motion sketch trajectories, and two new constraints. The first is a derivative constraint on the ground contact forces, which helps prevent excursions of the contact-force spline outside of friction cones. The second is a leg-length constraint which uses multiple planes (Figure 8), as we found the use of body-oriented box constraints to be problematic (See Appendix D.1 for details). The planner optimizes the contact forces as well as the CDM trajectory, with the equation of the motion of CDM as a constraint which describes the relationship between the contact forces and 6-DOF CDM motion. The footstep locations in the motion sketch are also further optimized to find more physically feasible locations. The CDM motion evolves over time via the multiple contact forces and contact locations, representing the limb endpoints. The contact forces, which are constrained to be within the friction cones, can generate highly dynamic motions with long flight phases and rapid rotations, as needed.

Optimization variables. The initial state and final state of the CDM $(\mathbf{p}_{\text{pend}}, \mathbf{R}_{\text{pend}})$ and the position and time/duration for each contact point (end-effector) $(\mathbf{p}_{i,j}, \{t_{i,j}\}, \{d_{i,j}\})$ are given as the motion sketch input, and the total duration T is experimentally chosen in advance ranging from 1 to 1.2 locomotion cycles. From this input, the COM position and orientation, and contact location and force for the planning horizon are optimized.

The CDM motion and the time-varying contact forces are represented as continuous trajectories using a sequence of cubic Hermite splines. Adjacent spline segments share the position and velocity keys at every connection point to satisfy C^1 continuity without increasing number of variables and constraints. Thus, the optimization variables include the position and orientation of the CDM and their time derivative $(\mathbf{q}, \dot{\mathbf{q}})$, and the contact force and its time derivative $(\mathbf{f}, \dot{\mathbf{f}})$ at the end points of each spline segment, as well as the

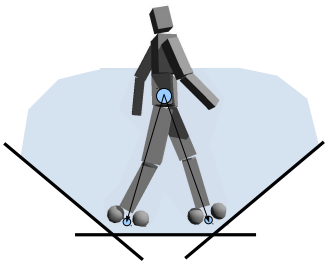


Fig. 8. The leg-length constraints. We use at most three planes perpendicular to the sagittal plane to constrain the centers of the feet within reach from the COM.

contact positions of each limb in the horizon ($\mathbf{p}$). See Appendix D.2 for details.

Constraints. The physical correctness of the CDM motion is guaranteed by constraints encompassing: the equations of motion, constraints for contact forces, constraints for contact positions, initial and final conditions, and C^2 continuity. See Appendix D.3 for details of the following constraints.

Equation of motion. The equation of motion is written as:

$$\mathbf{M}\ddot{\mathbf{q}}\left(\mathbf{q},\dot{\mathbf{q}}\right)+\mathbf{b}\left(\mathbf{q},\dot{\mathbf{q}}\right)=\mathbf{J}_c\left(\mathbf{q},\mathbf{p}\right)^T\mathbf{B}\left(\mathbf{p}\right)\mathbf{f}, \tag{10}$$

where $\ddot{\mathbf{q}}$ is CDM acceleration calculated as finite difference and $\mathbf{b}$ represents Coriolis, centrifugal and gravitational force. The inertia matrix $\mathbf{M}$ can be obtained by collecting the whole-body inertia matrix either from the reference motion or using a constant inertia matrix for every key-frames. $\mathbf{J}_c\left(\mathbf{q},\mathbf{p}\right)$ contains the Jacobian matrices that relate the global velocities at the contact points to the generalized velocity of the base of the CDM. $\mathbf{B}\left(\mathbf{p}\right)$ contains the basis vectors for the friction cones.

Constraints for contact forces. The ground contacts are modeled as:

$$\left|\mathbf{T}\left(\mathbf{p}\right)\mathbf{f}\right|\leq\mu\mathbf{N}\left(\mathbf{p}\right)\mathbf{f}, \tag{11}$$

$$\mathbf{0}\leq\mathbf{N}\left(\mathbf{p}\right)\mathbf{f}\leq\mathbf{f}_{\max}, \tag{12}$$

$$\left\|\dot{\mathbf{f}}\right\|\leq\epsilon_{\mathrm{f}}, \tag{13}$$

which imply the contact forces must lie within the linearized friction cones (Equation 11), and the vertical ground reaction force should be smaller than a threashold $\mathbf{f}_{\max}$ but should not pull the ground (Equation 12). $\mathbf{T}\left(\mathbf{p}\right)$ contains the basis vectors for the tangential contact plane, and $\mathbf{N}\left(\mathbf{p}\right)$ contains the contact normal vectors. μ is the friction coefficient, which is set to 1.0 for all experiments. The friction cone constraints are enforced for the key-frames of the splines representing contact forces. To prevent contact forces from excessive excursions outside of the friction cones, the magnitude of the contact force derivatives should be smaller than a threshold ϵ_{f} (Equation 13).

Constraints for contact positions. The contact positions are constrained as:

$$\forall i,\forall j;\ \mathbf{N}_{\mathrm{l}}\cdot\left(\mathbf{q}_{\mathrm{t}}-\mathbf{p}_{i,j}\right)\geq\mathbf{l}_{\max}, \tag{14}$$

$$\mathbf{p}_y=h_{\mathrm{terrain}}\left(\mathbf{p}_x,\mathbf{p}_z\right), \tag{15}$$

$$\forall i;\ \mathbf{p}_{i,0}=\mathbf{p}_{i,\mathrm{c}}^{\text{last-cdm}}, \tag{16}$$

$$\forall i,\forall j\in[1,n_{\text{contact}}^{i}];\ \left\|\mathbf{p}_{i,j}-\mathbf{p}_{i,j}^{0}\right\|\leq\epsilon_{\mathrm{p}}. \tag{17}$$

Equation 14 implies the approximated leg-length constraint, which constrains the contact positions to be within reach from the COM of the character. This spherical constraint is approximated using several planes. $\mathbf{N}_{\mathrm{l}}$ contains the normal vectors for the planes, $\mathbf{q}_{\mathrm{t}}$ indicates the translational components of $\mathbf{q}$, and $\mathbf{l}_{\max}$ is the maximum distance from the COM that the foot can reach. In our experience, three planes were enough for our purposes (Figure 8). Equation 15 constrains the height of the contact position $\mathbf{p}_y$ to be equal to the height of the terrain on which it is located $h_{\mathrm{terrain}}\left(\mathbf{p}_x,\mathbf{p}_z\right)$. Equation 16 constrains the 0-th footstep location $\mathbf{p}_{i,0}$ to be equal to the corresponding footstep location determined from the last CDM trajectory planning $\mathbf{p}_{i,\mathrm{c}}^{\text{last-cdm}}$.

Initial and final condition, C^2 continuity. Additional constraints are defined as:

$$\mathbf{q}\left(0\right)=\mathbf{q}_0,\dot{\mathbf{q}}\left(0\right)=\dot{\mathbf{q}}_0,\mathbf{q}\left(T\right)=\mathbf{q}_T,\dot{\mathbf{q}}\left(T\right)=\dot{\mathbf{q}}_T, \tag{18}$$

$$C_2\left(\ddot{\mathbf{q}}\left(\mathbf{q},\dot{\mathbf{q}}\right)\right)=0, \tag{19}$$

to set the initial and final state of the CDM to the corresponding COM positions and pendulum orientations and their derivatives in the motion sketch (Equation 18) and to additionally satisfy C^2 continuity condition for the CDM trajectory splines (Equation 19). The reason is that a discontinuity in the acceleration implies a discontinuity in the contact forces, which contradicts our continuous spline-based formulation.

Objective function. The objective function E_{CDM} measures the difference between the desired CDM trajectory and actual CDM trajectory:

$$E_{\mathrm{CDM}}=\left\|\mathbf{q}-\bar{\mathbf{q}}\right\|^2+w_d\left\|\dot{\mathbf{q}}-\dot{\bar{\mathbf{q}}}\right\|^2, \tag{20}$$

where the desired CDM trajectory $\bar{\mathbf{q}}$ is defined by sum of the pendulum trajectory and the user-defined horizontal oscillatory displacement of the COM, and $\dot{\bar{\mathbf{q}}}$ is its time-derivatives. See Appendix D.4 for details of the desired CDM trajectory $\bar{\mathbf{q}}$.

Nonlinear programming formulation. The CDM optimization is formulated as a nonlinear programming using the objectives and constraints stated above:

$$\underset{\mathbf{q},\dot{\mathbf{q}},\mathbf{f},\dot{\mathbf{f}},\mathbf{p}}{\text{minimize}}\quad E_{\mathrm{CDM}}, \tag{21}$$

$$\text{subject to}\quad \text{Equation 10–19}, \tag{22}$$

We use IPOPT to solve the large-scale nonlinear optimization as it provides efficient support for sparse matrices. The NLP solver requires not only the constraint equations but also their first derivatives (sparse Jacobian matrices). Analytical derivatives are used for all constraints except for the terrain constraints, for which we use numerical derivatives. Note that this optimization is defined in a high dimensional space, is highly nonlinear, and thus prone to local

minima. It can easily lead to bad local minima without a good initial guess. The use of a physically plausible motion sketch as an initial solution and in the objective function helps improve the robustness of the optimization, allows for more efficient planning, and better quality results.

5.2 CDM Forward Dynamics

We integrate the ability to exploit CDM forward dynamics into our system in order to allow a character to respond interactively to unanticipated external forces.

The footstep locations, timing, duration, and contact forces are determined during the footstep and CDM trajectory planning. In the CDM forward dynamics step, it is assumed that these contact plans remain unchanged, but that the CDM trajectory is modified due to external forces. This assumption might seem somewhat restrictive, but it results in natural responses to external pushes. For example, if a downward force is applied to the head of a character, the character's COM is lowered to reflect the downward force while maintaining the contact forces. A user-specified external force can be applied to any point on the CDM. The sum of the external force and the NLP solver contact forces then govern the motion of the CDM until the next planning interval.

The pendulum trajectory and footstep plans are recomputed at the next footstep or the start of the next locomotion cycle, in order to recover from the external push and follow the user-specified desired direction and speed once again. This simple approach exhibits the desired recovery balancing behavior. Excessive external forces can lead to a situation where the COM position becomes unrealistically far from the planned contact positions. However, in our experiments, this was not problematic because such a strong force results in a fall, as reflected by the NLP solver failing to find a feasible solution.

6 FULL-BODY MOTION GENERATION

The rough sketch of overall body movement and footstep plan is computed by the motion sketch generator. The CDM plan generator refines the sketch with more detailed behaviors. As the generated CDM plan is guaranteed to be physically correct, the remaining step is just converting the CDM plan to the full-body motion, which is performed by our momentum-mapped inverse kinematics solver.

6.1 Momentum-mapped Inverse Kinematics

The goal of a momentum-mapped inverse kinematics solver is to compute the character's joint angles ($\mathbf{x}$) to position an end-effector on a planned footstep location while matching the planned CDM configuration and its time-derivative. Unlike a conventional inverse kinematics solver, our solver takes as input the resulting pose of the previous frame as well as the reference pose to consider both joint angles and velocities.

The momentum-mapped inverse kinematics solver minimizes the following cost function E_{IK}:

$$\begin{aligned} E_{\mathrm{IK}}(\mathbf{x}) = &\sum_k i_k \left\| \mathbf{y}_k(\mathbf{x}) - \overline{\mathbf{y}}_k \right\|^2 \\ &+ w_{\mathrm{g}} \left\| \mathbf{I}_{\mathrm{F}}^{-1} \mathbf{J}_{\mathrm{M}} (\mathbf{x} - \overline{\mathbf{x}}) \right\|^2 + w_{\mathrm{m}} \left\| \mathbf{I}_{\mathrm{F}}^{-1} \mathbf{J}_{\mathrm{M}} \dot{\mathbf{x}} - \dot{\boldsymbol{\Theta}}^{\mathrm{CDM}} \right\|^2 \\ &+ w_{\mathrm{p}} \left\| v_{\mathrm{p}} - \mathbf{N}_{\mathrm{p}} \mathbf{J}_{\mathrm{p}} \dot{\mathbf{x}} \right\|_{\oplus}^2 + w_{\mathrm{v}} \left\| \dot{\mathbf{x}} - \dot{\overline{\mathbf{x}}} \right\|^2 + w_{\mathrm{r}} \left\| \mathbf{x} - \overline{\mathbf{x}} \right\|^2, \end{aligned} \quad (23)$$

where $\mathbf{x}$ is the joint angles which represent the full-body pose, $\dot{\mathbf{x}}$ is the joint velocities obtained by the backward difference. The desired joint angles $\overline{\mathbf{x}}$, which represent the desired pose, is obtained from a reference pose by rigidly transforming it to match the current CDM position and orientation obtained by sampling the spline $(\mathbf{q}, \dot{\mathbf{q}})$ for the planned CDM trajectory at the current time t, and the desired joint velocities $\dot{\overline{\mathbf{x}}}$ is obtained from its backward difference.

The first term measures the error between $\mathbf{y}_k(\mathbf{x})$, the k-th end-effector position computed from the full-body pose $\mathbf{x}$, and $\overline{\mathbf{y}}_k$, the desired k-th end-effector position from the planned footstep locations.

The second term aims to match the planned CDM configuration from $\overline{\mathbf{x}}$ and the CDM configuration from $\mathbf{x}$, which adapts the idea of the momentum-based geometric mapping [Kwon and Hodgins 2017]. $\mathbf{I}_{\mathrm{F}}$ is the composite rigid body inertia (CRB inertia) matrix of the full-body character, and momentum Jacobian $\mathbf{J}_{\mathrm{M}}$ relates joint velocities to generalized centroidal momentum.

The third term measures the error between the generalized velocities of the planned CDM and the CDM from estimated $\mathbf{x}$. $\dot{\boldsymbol{\Theta}}^{\mathrm{CDM}}(\mathbf{q}, \dot{\mathbf{q}}, t)$ is the generalized velocity of the planned CDM.

The fourth term is used only when an external force is applied to a body part distant from the COM, and it constrains the application point to move faster than v_{p} along the push direction $\mathbf{N}_{\mathrm{p}}$. v_{p} is a heuristic velocity threshold that is given proportional to the magnitude of the external force. Roughly speaking, it plays a similar role to the velocity cone in LCP-based collision processing. $\|\cdot\|_{\oplus}$ vanishes when the inner part $\cdot$ is negative. $\mathbf{J}_{\mathrm{p}}$ is the Jacobian for the application point. This term uses the redundant DOFs to make the result more visually pleasing. Without this term, the full-body character response to the external force using the entire body without any compliance.

The last two terms are regularization terms to keep the result close to the desired full-body pose $\overline{\mathbf{x}}$ and its derivative $\dot{\overline{\mathbf{x}}}$.

The scalar $w_{\mathrm{g}}, w_{\mathrm{m}}, w_{\mathrm{p}}, w_{\mathrm{v}}$, and w_{r} are the weights for the objective terms, respectively. The weights w_{r} and w_{v} for the regularization terms are set as relatively small numbers between 10^{-2} and 10^{-3}, while other weights are close to one.

We formulate the inverse kinematics problem as an optimization problem. This problem can be efficiently solved numerically using an L-BFGS-b solver with an analytic gradient function even though our search space can be large, for example, 32 DOFs for a human character. See Appendix E for details of the formulation.

7 LEARNING FRAMEWORK FOR CENTROIDAL DYNAMICS

Although our motion generation system described in Section 4–6 can generate final full-body motions at interactive rates, the CDM

trajectory planning can take too much time for smooth real-time performance and sometimes fails to find the feasible solution due to the complexity of the nonlinear programming. To tackle this issue, we propose to train a deep neural network (DNN) for solving the CDM optimization problem efficiently, as well as robustly, by training on feasible samples. By using the results of the pendulum planner and the footstep planner as input, various difficulties of existing motion regression schemes are effectively avoided. For example, the network does not need to determine when to make footsteps because contact timing is already fixed. Foot locations are also easy to predict because the pendulum trajectory provides good initial guidance.

The trained network predicts the CDM plan and full-body motion for the CDM planning horizon (1–1.2 locomotion cycles) at every footstep (for biped characters) or every locomotion cycle (for other characters).

7.1 Network Input and Output

The inputs to the network are twofold: the initial model state and future motion information. The former consists of the initial CDM state (6-DOFs position and velocity $\mathbf{q}_0, \dot{\mathbf{q}}_0$), initial feet positions ($\mathbf{p}_0$), and the initial full-body pose ($\mathbf{x}_0$), which are the last predicted ones that corresponds to the starting frame of the current prediction horizon. The latter consists of sets of features sampled from the motion sketch and terrain geometry, such as the pendulum configuration ($\mathbf{p}_{\text{pend}}$, $\mathbf{R}_{\text{pend}}$), phases for all limbs (ψ_i, see Appendix E for details), terrain heights around the cart. The number of sample times for the planning horizon we use is 10 for all experiments. The dimension of the network input depends on the character model. For example, the input has 105 dimensions for Humanoid scenarios. See Appendix F.1 for details of the network input.

The outputs from the network are the CDM states ($\mathbf{q}_{\text{out}}, \dot{\mathbf{q}}_{\text{out}}$), feet positions ($\mathbf{p}_{\text{out}}$), and full-body poses ($\mathbf{x}_{\text{out}}$), uniformly sampled over the planning horizon. The number of samples is experimentally set as 30, and the final outputs are reconstructed from the samples using the uniform cubic interpolating splines. To make sure the predicted full body motion and CDM plan are consistent, all network outputs (CDM states, feet positions, and full-body poses) are encoded locally to the input pendulum frame at the corresponding time. See Appendix F.2 for details of the network output.

Note that, unlike most learning-based approaches, our network takes future information for variable-length duration. The window size is set same as the CDM planning horizon which is defined using locomotions cycles. The output of our network is a full-body motion segment and a CDM plan for the planning horizon, not a single full-body pose at the next frame which is the commonly used scheme by the existing learning-based approaches. Using our design of the network inputs, this per-segment network outperforms the traditional per-frame network. The detailed comparison is given in Section 8.

7.2 Network Structure

We use a single neural network with three hidden layers, which generates a motion that corresponds to the CDM planning horizon at once. Specifically, given input features $\mathbf{x} \in \mathbb{R}^{n_{\text{i}}}$ and output parameters $\mathbf{y} \in \mathbb{R}^{n_{\text{o}}}$, we build a neural network $\mathbf{\Phi}$ as follows:

$$\mathbf{\Phi}(\mathbf{x}) = \mathbf{W}_3 \text{ELU}(\mathbf{W}_2 \text{ELU}(\mathbf{W}_1 \text{ELU}(\mathbf{W}_0 \mathbf{x} + \mathbf{b}_0) + \mathbf{b}_1) + \mathbf{b}_2) + \mathbf{b}_3, \tag{24}$$

where the parameters of the network are $\mathbf{W}_0 \in \mathbb{R}^{50 \times n_{\text{i}}}$, $\mathbf{W}_1 \in \mathbb{R}^{12 \times 50}$, $\mathbf{W}_2 \in \mathbb{R}^{200 \times 12}$, $\mathbf{W}_3 \in \mathbb{R}^{n_{\text{o}} \times 200}$, and the corresponding biases $\{\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\}$. The exponential rectified linear function (ELU) is used as the activation for generating smooth outputs [Clevert et al. 2016]. All the hyper-parameters are experimentally chosen to achieve both good generalization and accurate fitting. Note that we used a small number of hidden units ($n_h = 12$) in the second hidden layer. This is probably the most important hyper-parameter which is necessary to avoid the overfitting, together with a dropout [Srivastava et al. 2014] applied to the input layer with retention probability of 0.99. We used the mean squared error loss function.

7.3 Network Training

In the offline learning stage, each generator creates the outputs for random user inputs, and use them to extract the training data for the learning framework. Specifically, at every planning step, the desired speed is changed with the probability of 0.3 by adding a random value from a normal distribution ($\sigma = 1.0$, $\mu = 0$) to the previous desired speed. Similarly, desired direction is modified using a random value from a normal distribution ($\sigma = 0.5$, $\mu = 0$). Data generation restarts after saving previous results whenever an error occurs from the NLP solver or the character goes out of the terrained map.

A deep neural network is trained for a relatively short time for a single character model to output the CDM plan and corresponding full-body pose of the character for input motion sketches. For example, the network for the human character can be distilled from 39350 frames of randomly generated motion in about 30 minutes including both generation and training.

7.4 Online Full-body Motion Generation

At every planning step, the motion sketch generator produces the motion sketch for the user-specified parameters. The inputs for the neural network is created from the motion sketch (for the future motion information) and the last prediction of the network (for the initial model state), then the final CDM motion and full-body motion for the CDM planning horizon are generated immediately from the trained neural network. The generated motions are used as the inputs for the momentum-mapped inverse kinematics solver until the next planning step, and the motions for the next CDM planning horizon are generated from the network again.

The CDM trajectory predicted from the DNN can be fed into the CDM forward dynamics simulator, with the aggregate contact forces computed, from the predicted CDM trajectory by solving the inverse dynamics problem which is trivial for the CDM. This means that the DNN can handle online perturbations in the same way as the CDM planner, by assuming that the aggregate contact forces do not change until the next planning step.

The full-body motion from the DNN is generally smooth and contains very little foot-skating. However, for some extreme cases, the motion from the previous planning and the current planning might be slightly less smooth. To reduce this artifact, we simply

stitch these motions or blend them for the overlapped interval. This might cause a small amount of foot-skating, which can be later removed using the momentum-mapped inverse kinematics solver which is executed every frame, provided with the blended motion as a good initial guess.

8 RESULTS

We evaluate the method on the four character models with a large variety of motions and environments (See Appendix A for details of the four character models: Humanoid, Luxo, ANYmal, and Cassie). All the results are best seen in the video that accompanies this paper. Each result is generated either using the CDM-based motion generation system or using the learning-based motion generation system.

The list of scenarios and timing statistics are summarized in Table 4 in Appendix. All timing statistics are obtained using an ultra-portable laptop, equipped with a Intel i7-8550U. The pendulum trajectory is generated at 30 Hz. The spline segments for CDM optimization are densely placed in time (0.16 s for slow walks). For each scenario, the time step is increased to speed up the calculation if there is no difference in motion quality and robustness. For *Learning framework*, all networks are trained using PyTorch in mini-batches of size 32, and in 400 epoches.

The process of adjusting the parameters or weights when necessary is intuitive and predictable, and thus desirable results are quickly realized, particularly because motions are synthesized at interactive rates. A wide range of parameters and weights work robustly, yielding stylistic variations. See Appendix H for details of experimental parameters.

Walking and running motions. We use a two-second-long walking motion and a 1.5-second-long running motion for these experiments. Note that our system smoothly generates the abrupt changes of moving direction or speed which are not in the reference motions, as demonstrated in Humanoid-SharpTurn (Figure 10(a)) and Humanoid-RunVaryingSpeed (Figure 10(b)). Specifically, the Humanoid walking at 4.14 km/h makes 180 degrees turns or a full-stop only in one or two steps. The running Humanoid accelerates from 20 km/h to 35 km/h in about three seconds. The Humanoid also can run on irregular terrain, as demonstrated in Humanoid-TerrainRun. For these Humanoid scenarios, the desired direction or speed is controlled by the user and the gait parameters and the COM oscillatory displacement are extracted from the corresponding reference motion. The magnitude of the COM oscillation is reduced in inverse proportional to the desired speed as a fast speed with a large oscillation tends to make the simulation unstable. To produce a natural motion pose at various speeds, for both walking and running motions, the input pose to the momentum-mapped inverse kinematics solver at every frame is exaggerated in proportion to the stride length, using the standard motion editing scheme described in [Bruderlin and Williams 1995]. Our system effectively generates a wide range of stride length or duration, as demonstrated in Humanoid-WalkAndStop and Humanoid-RunVaryingSpeed, based only on the user-specified desired speed while using the same constant desired values extracted from the reference motion.

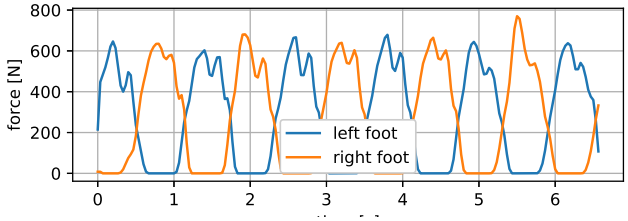


Fig. 9. Vertical contact force profiles for the Humanoid walking at 1.1 m/s.

The generated contact force profile for the Humanoid walking is similar to the real person. Figure 9 shows the vertical contact force for the generated walking motion at 1.1 m/s. Double-peaks are observed in the plot, which is the characteristic of human walking. The design of the contact force trajectory as a spline in the CDM trajectory planner enables such a realistic contact force profile. This result is actually impressive because it is generated from the simplified model, the CDM, not from the full-body dynamics of the Humanoid model.

Our system can also generate a variety of locomotion skills for other character models with different topological structures. In Luxo-Terrain and ANYmal-Terrain, the Luxo and ANYmal model move freely over uneven terrain only by specifying the desired direction. Their desired speed and gait parameters are given constant. The horizontal COM oscillatory displacement is not given for all Luxo scenarios, because the Luxo does not need such oscillation as it has only a sigle limb. It does not given to all ANYmal scenarios as well, because the ANYmal generates the horizontal oscillation by restricting the deviation from the planned footstep locations during the CDM trajectory optimization. Otherwise, the calculated footstep positions easily generate a undesirable pose for this multi-limb character.

Learning framework. Our learning framework is so general that it can be applied to any kind of character model and motion. We experiment on the Humanoid (biped with reference motions) and ANYmal (quadruped without any reference motion) as representative models. The detailed statistics for these scenarios are provided in Table 1.

For Humanoid-DNN, a single network is used for all the human motions, that is for both walking and running, and for both left-foot initiated and right-foot initiated segments. The user can change the desired motion type as well as the desired direction and speed. If the desired motion type is changed, the desired speed is automatically adjusted to be within the speed range of that motion. The actual motion type, which is a part of the network input, is changed when the character's actual speed comes into the speed range of that motion. The gait parameters extracted from each reference motion are applied to the corresponding motion type in both online simulation and offline training, and the COM oscillatory displacement is also extracted from the reference motions and its maginitude is adjusted in consideration of the desired speed as in other scenarios. As shown in Table 1, the training dataset for this scenario containing 39350 frames is fully automatically obtained in 26 minutes and the network is trained for about 4 minutes.

Table 1. The detailed statistics for *Learning framework* scenarios. Prediction time is normalized so that the output corresponds to one second.

	Humanoid-DNN	ANYmal-DNN
# of training data frames	39350	11940
# of training data segments	2722	577
data generation time	26 m	98 m
training time	4 m 4 s	63 s
prediction time	2.9 ms	1.56 ms
network input dim.	105	132
network output dim.	1500	1440

Similarly, a single network is trained for ANYmal-DNN. The user can only change the desired direction and speed in this scenario. The same set of gait parameters pre-defined by the user is used in both online simulation and offline training. The training dataset for this scenario containing 11940 frames is fully automatically obtained in 98 minutes, and the network is trained for about 1 minute.

We conducted a comparison experiment between terrain-walking Humanoids with and without using the learning framework. As shown in the accompanying video, the motion generated using the learning framework shows almost the same quality as the motion generated using the CDM-based generation system, but the online generation speed is much faster (about 8 times) for the learning-based framework.

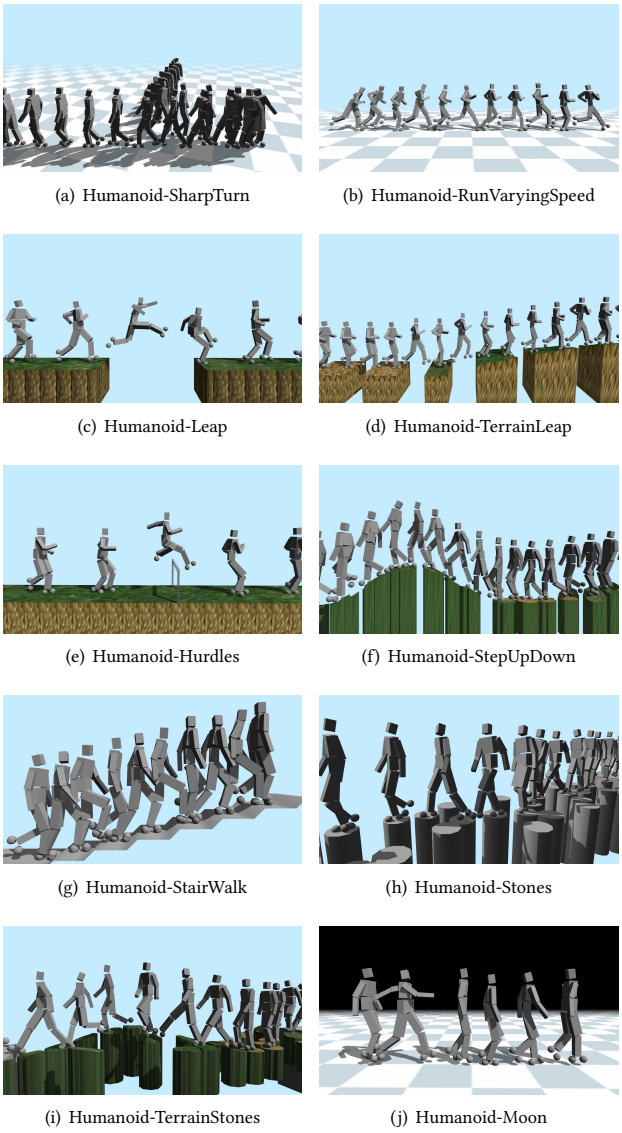
For ANYmal-DNNPush, the network trained for ANYmal-DNN is used to predict CDM states and the contact force is computed using an inverse dynamics solver. They are fed into the CDM forward dynamics along with the user-specified external forces to generate a responsive CDM motion, as described in Section 7.4.

Environmental variations. The Humanoid model can run while leaping over gaps at various speeds. A single forward jumping motion is used for these experiments. It robustly leaps over 3 m-wide gaps installed on a flat ground at random intervals of 2–2.5 m at speed of 5 m/s (Humanoid-Leap, Figure 10(c)), and gaps of 0.1–1 m width installed on terrain at random intervals of 1–1.8 m at speed of 2.68 m/s (Humanoid-TerrainLeap, Figure 10(d)). The gaps are modeled as the obstacle constraints (Equation 9) not to make footsteps in the gaps during footstep planning.

We also generate motions for hurdle runs (Humanoid-Hurdles, Figure 10(e)). The vertical COM movement necessary for jumping over the hurdles is created using an invisible 3 m-wide gap placed at each hurdle position, with a running speed of 5 m/s. A single key-pose at the peak height is used to define the typical leaping pose of hurdle runs. The key pose is first blended with the pose from the reference running motion in order to generate a smooth transition between running and jumping based on the motion phase, and then used as input by the momentum-mapped inverse kinematics solver.

As described in Appendix C.6, the footstep locations should be determined before the cart trajectory for the stepping stones cases. We demonstrate this variant of motion sketch generation with four scenarios, Humanoid-StepUpDown (Figure 10(f)), Humanoid-StairWalk (Figure 10(g)), Humanoid-Stones (Figure 10(h)), and Humanoid-TerrainStones (Figure 10(i)). In these scenarios, the footstep planner optimizes the number of stepping times for each stone or the sequence of stones to step on in the irregularly-placed stones or regularly-spaced stairs environments. In addition, we compare the results of characters with different moving speeds in the same environment. Our derivative-free optimization for footstep planning allows interesting behaviors such as stepping on a different sequence of stones for different moving speed, which could not be achieved with a derivative-based optimizer.

We also demonstrate reduced gravity locomotion (Humanoid-Moon, Figure 10(j)). Reducing the gravitational accelaration to the moon's level (1.6 m/s^2) and specifying the desired stride duration



(a) Humanoid-SharpTurn (b) Humanoid-RunVaryingSpeed

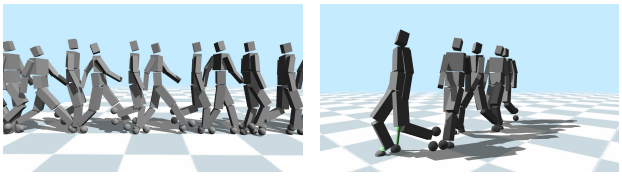
(c) Humanoid-Leap (d) Humanoid-TerrainLeap

(e) Humanoid-Hurdles (f) Humanoid-StepUpDown

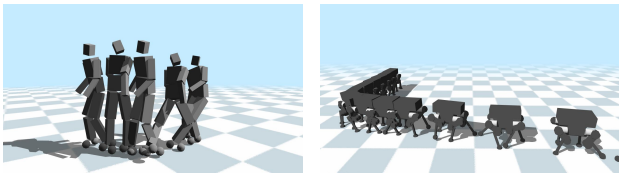
(g) Humanoid-StairWalk (h) Humanoid-Stones

(i) Humanoid-TerrainStones (j) Humanoid-Moon

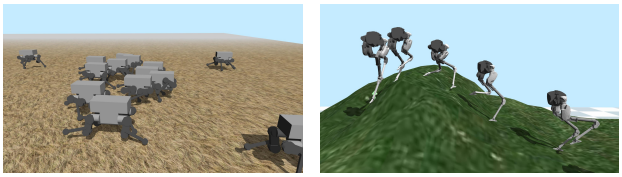
Fig. 10. *Walking and running motions* and *Environmental variations* scenarios.



(a) Humanoid-VaryingStride (b) Humanoid-VaryingLatFoot

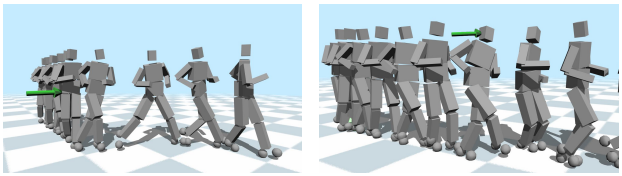


(c) Humanoid-Spin (d) ANYmal-Gaits

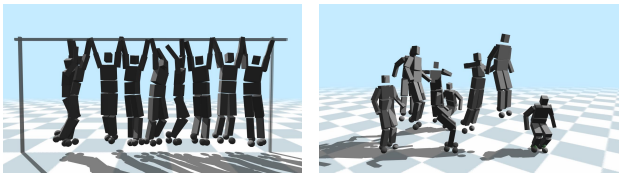


(e) ANYmal-Rush (f) Cassie-Style3

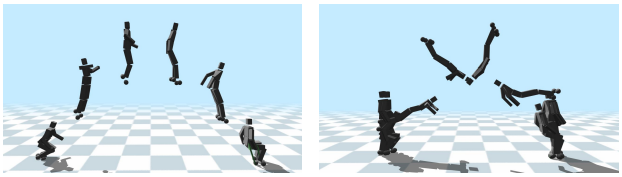
Fig. 11. *Locomotion style variations* scenarios.



(a) Humanoid-Push (COM, 600 N, 0.2 s) (b) Humanoid-Push (head, 200 N, 0.2 s)



(c) Humanoid-Monkeybars (d) Humanoid-RandomJumps



(e) Humanoid-SuperJumps (f) Humanoid-SuperFlips

Fig. 12. *External pushes* and *Monkeybars, jumps, and superhuman jumps* scenarios.

four times longer, our system generates natural locomotion that is likely to be observed on the moon.

Locomotion style variations. The gait parameters or COM horizontal oscillation can be adjusted to generate various styles of locomotion. In Humanoid-VaryingStride (Figure 11(a)), the desired stride length is changed from 0.5 m to 1.1 m to generate humanoid locomotion with varying stride. In Humanoid-VaryingLatFoot (Figure 11(b)), the COM oscillatory displacement is edited to adjust lateral foot placements. It is edited to move the COM in the direction of the stance leg in the lateral plane to generate catwalk-style walking, and in the opposite direction to keep the legs wide apart while walking.

The style of locomotion can be changed significantly by changing the reference motion, as a stylistic guide for the momentum-mapped inverse kinematics solver. The Humanoid exhibits four different walking styles with the corresponding reference motions (Humanoid-ArmWideOpen, Humanoid-LeanedBack, Humanoid-Sneaky, Humanoid-March). The gait parameters and COM oscillatory displacement are extracted from the corresponding reference motion. Additionally, the Humanoid can reproduce totally different locomotion style, Humanoid-Spin (Figure 11(c)), with a spinning reference motion.

For the ANYmal quadruped model, all five different types of gaits (walk,trot,canter,pace and gallop) are generated only by specifying different desired speeds from 1 m/s to 11 m/s while using the same set of objective functions and coefficients, and the pre-defined gait parameters (ANYmal-Gaits and ANYmal-Rush, Figure 11(d) and 11(e)). Switching between gait types occurs smoothly as the desired speed changes. In ANYmal-Rush, the many big-ANYmal models rush at various speeds. The different types of gaits used by the small ANYmal (walk,trot,canter,pace) and the big ANYmal (walk,trot,canter,gallop) emerge from the difference in body length. See Appendix G for more details.

For the Cassie model, three different style variations of walking are demonstrated (Cassie-Style1–3, Figure 11(f). These variations were generated by changing the gait parameters such as the ratio of the stance phase to the flight phase, the desired cycle duration and the desired speed. The same set of manually-tuned COM oscillatory displacement is used for all Cassie scenarios to generate natural locomotion.

External pushes. Integrating CDM forward dynamics into our system allows the character models to respond to unexpected external forces. In Humanoid-Push, the Humanoid model can recover from multiple 100 N–800 N pushes of duration 0.2 s applied at the COM (Figure 12(a)), and multiple 200 N pushes of duration 0.2 s applied to different body parts (Figure 12(b)). As the CDM forward dynamics integration does not make any difference if there is no additional push force, we apply it only to Humanoid-Push and ANYmal-DNNPush.

Monkeybars, jumps, and superhuman jumps. This set of scenarios demonstrate that the full-body motion can be generated without using the motion sketch generator in cases where it is stable enough without using it.

For Humanoid-Monkeybars (Figure 12(c)), we provide a rough sketch of the CDM trajectory, which is manually key-framed as shown in the accompanying video, and a manually designed contact

plan to the CDM planner. The CDM planner converts this rough motion to the physically correct CDM motion with the contact forces. The contact forces are calculated so that the monkey bars "pulls" the character to realize the desired CDM trajectory. The momentum-mapped inverse kinematics solver takes reference poses from a walking motion as input, and produces a completely different, but still plausible, motion of the character navigating the monkey bars by placing the end-effectors of the contact limbs, the hands, on the contact positions on the monkey bars.

In the jumping scenarios, the CDM trajectory extracted from a captured jumping motion is procedurally edited, and then used as input to the CDM planner. For the editing, we simply segment the CDM trajectory into three consecutive segments, the second of which contains the flight phase. The second segment is then edited for the desired jump height and direction. For the height adjustment, the COM path during the flight phase is reconstructed using a parabolic curve consistent with the law of physics. For direction adjustment, the forward direction during the flight phase is edited using a constant angular velocity. After smoothly stitching the second segment with other segments, the resulting sequence is used as an input motion sketch. Using this scheme, a randomly jumping motion (Humanoid-RandomJumps, Figure 12(d)) can be easily generated by randomly choosing the desired direction at each jump. We replace the soft constraint for end-effector positions in the momentum-mapped inverse kinematics solver (the first term of Equation 23) with a hard constraint to prevent foot sliding artifacts.

This simple scheme works well even with a seven-fold increase of the jump duration to yield a superhuman motion (Humanoid-SuperJumps, Figure 12(e)). The orientation of the extracted CDM trajectory is also edited to rotate the body while jumping (Humanoid-SuperFlips, Figure 12(f)). To generate these extreme behaviors, the hard constraint on the maximum vertical ground reaction force (Equation 12) is turned off; without this, the CDM optimization (unsurprisingly) fails to find a feasible solution.

Statistics for computation. As can be seen in Table 4 in Appendix, many motions can be generated in real-time ($T_{\text{comp}}/T_{\text{clip}} < 1$) even without using the learning framework. However, it takes a much longer time to generate some motions with a faster speed, short stance phases or many limbs. For example, Humanoid-Hurdles (5 m/s) and ANYmal-Gaits (0-10 m/s) spend 15.5 s and 19.7 s for generating one second full-body motion, respectively. This tendency can also be observed in Table 2, which shows the break down of the computation time and other information for generating locomotion of four character models on a flat ground. This is probably because more splines and variables are used for a higher speed motion, and increasing flight duration makes it more difficult to find the feasible solution in the CDM trajectory optimization due to the longer unactuated phase. The more number of limbs increases ambiguity in that many different solutions can lead to the same COM trajectory, and thus more inequality constraints become active. It also takes relatively longer time when the footstep locations need to stay close to the specified locations, such as Humanoid-Monkeybars or Humanoid-StairWalk, which is probably because it is more difficult to find the solution in a narrow region.

Table 2. Statistics for generating locomotion of four character models on a flat ground. Walk: Humanoid walking at 1.15 m/s. Run: Humanoid running at 1.76 m/s. Fast run: Humanoid running at 7 m/s. Luxo: Luxo moving at 1.36 m/s. ANYmal: ANYmal walking at 1 m/s. In the upper part of the table, average computation time for generating one second of final motion is measured. Total time includes all the necessary computation time except the rendering time.

	Walk	Run	Fast run	Luxo	ANY-mal
generation time for ...	average computation time				
motion sketch	0.07s	0.10s	0.18s	0.084s	0.82s
CDM plan	0.16s	0.13s	1.08s	0.82s	14s
full-body motion	0.23s	0.23s	0.23s	0.10s	0.16s
total	0.46s	0.46s	1.49s	0.93s	15s
	statistics related to sketch planning				
planning horizon	2.11s	1.38s		2.79s	2.56s
	statistics related to CDM planning				
planning horizon	0.80s	0.40s	0.21s	0.79s	0.62s
	1.2 half cycle			1 full cycle	
# of variables	165	177	177	136	210
# of ineq. constraints	103	106	94	79	150
# of eq. constraints	83	92	104	84	64

Our learning framework effectively tackles these issues. Regardless of the speed or the number of limbs, our learning-based motion generator produces natural full-body motions online. The ANYmal walking motion can be genereated in real-time with learning, while it is about 15 times slower than real time without learning.

More experiments. We conduct additional comparison, ablation, and real-time experiments for our system (see Appendix I).

9 DISCUSSION

In this paper, we presented an interactive online MPC-based control system based on simplified dynamic models. The model is capable of producing a rich variety of terrain-adaptive motions for monopeds, bipeds, and quadrupeds, including jumps, banking runs on variable terrain, terrain leaps, monkey bars traversal, spin walks, dynamic push responses, and emergent quadruped gaits. These are achieved in a model-based fashion without any preprocessing. The online performance can be significantly improved using minimal offline learning. In support of these capabilities, we have introduced a three-level optimization process, emergent gait patterns, derivative-free foot placement and timing in support of stepping-stone problems, forward-dynamics CDM simulation for interactive motion response, and several other features.

Our approach has a number of practical considerations and limitations. The CDM model assumes that the inertia matrix is not a function of the state. We simply use a constant inertia matrix, but it is also possible to use a time-dependent inertia that might be obtained from motion capture data, for example. As a result, an inertia shaping strategy cannot be used for balancing or controlling the rotational speed.

Our momentum-mapped inverse kinematics solver can significantly modify the input reference pose to match the planned CDM configuration. However, it tends to use the upper-body rather than using arm motions because the upper body has a larger mass and inertia. In contrast, many human motions use large arm motions to initiate rotations or to assist balance recovery. We further note that the CDM model does not have joint information. The information needed for the momentum-mapped inverse kinematics comes from a user-supplied reference motion or keyframes. Thus the quality of the final motion reconstruction is dependent on reasonable quality keyframes or reference motion poses.

The CDM planning can fail in some circumstances, such as when a character approaches a steep slope too quickly. However, we observe that the learning-based motion generation can robustly extrapolate from the existing feasible examples used for training. In such cases, the physical accuracy and naturalness of the resulting motion are compromised, although we did not observe visible artifacts.

The CDM planner will naturally fail in extreme cases when some constraints cannot be met, for example, when an external force is too strong or the maximum vertical contact force constraint is not turned off for the super-jump examples. Over-constrained problems can be resolved via the allocation of additional spline segments or the removal of constraints. As future work, we wish to develop methods that automatically identify and remove constraints so as to still produce the most reasonable motions. This could include the automatic conversion of hard constraints to soft constraints.

We choose to fix contact timings instead of contact positions in the CDM planning, because of computational efficiency. The NLP solver IPOPT assumes a fixed sparsity structure of the input matrices and provides efficient support for sparse matrices. To allow changes in the planned footstep timing, however, the sparse Jacobian matrices for contact positions and forces become dense because of the possible changes in the key-frame dependency. Although we take into account collisions between the limbs during the footstep planning using soft-constraints, it remains possible for limbs to intersect during pushes or sharp turns.

There remain many directions for improvement. The CDM optimization can be improved in several ways. We wish to learn the coefficients for the objective function from example motions. The optimization time for our method currently has significant variation, ranging from 3× real-time to 15× slower than real-time, depending on the character and motion type. Although our learning framework is an online solution for motion synthesis, reducing the optimization time is still meaningful because it reduces the generation time for the training data. The solve time for the CDM planner is highly dependent on the plausibility of the initial solution. For dynamic walking or running motions, the high-level motion sketch provides a nearly-ideal initial solution, and thus the CDM planner is able to converge quickly. However, for a large gap that requires a leap, the initial COM trajectory needs to be modified significantly, resulting in a much longer solve time. Warm-starts that provide more plausible initial solutions from a pre-trained prediction model are an obvious possible improvement. We expect that state of the art data-driven methods, e.g., [Zhang et al. 2018], could be used. We wish to further explore the impact of the CDM planning horizon, which also has significant impact on the execution speed. In the case of quadruped walking, we use a full-cycle for easy implementation. However, it is possible that a half-cycle or less may produce similar quality of motion in many cases.

ACKNOWLEDGMENTS

The authors thank the anonymous reviewers for their many valuable comments. This work was partially supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (NRF-2019R1A4A1029800, NRF-2020R1A2C1012847, NRF-2019R1C1C1006778).

REFERENCES

Yeuhi Abe, Marco da Silva, and Jovan Popović. 2007. Multiobjective Control with Frictional Contacts. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 249–258.

Bernardo Aceituno-Cabezas, Carlos Mastalli, Hongkai Dai, Michele Focchi, Andreea Radulescu, Darwin G Caldwell, José Cappelletto, Juan C Grieco, Gerardo Fernández-López, and Claudio Semini. 2018. Simultaneous Contact, Gait, and Motion Planning for Robust Multilegged Locomotion via Mixed-Integer Convex Optimization. *IEEE Robotics and Automation Letters* 3, 3 (2018), 2531–2538.

Mazen Al Borno, Martin De Lasa, and Aaron Hertzmann. 2013. Trajectory optimization for full-body movements with complex contacts. *IEEE transactions on visualization and computer graphics* 19, 8 (2013), 1405–1414.

Mazen Al Borno, Michiel Van De Panne, and Eugene Fiume. 2017. Domain of attraction expansion for physics-based character control. *ACM Transactions on Graphics (TOG)* 36, 2 (2017), 17.

G. E. P. Box and G. C. Tiao. 1992. *Bayesian Inference in Statistical Analysis*. John Wiley & Sons, New York.

Armin Bruderlin and Lance Williams. 1995. Motion Signal Processing. In *SIGGRAPH*. 97–104.

Justin Carpentier, Steve Tonneau, Maximilien Naveau, Olivier Stasse, and Nicolas Mansard. 2016. A versatile and efficient pattern generator for generalized legged locomotion. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 3555–3561.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *arXiv:1511.07289 [cs]* (Feb. 2016). arXiv: 1511.07289.

Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. 2010. Generalized biped walking control. *ACM Transactions on Graphics* 29, 4 (2010), 130.

Stelian Coros, Andrej Karpathy, Ben Jones, Lionel Reveret, and Michiel Van De Panne. 2011. Locomotion skills for simulated quadrupeds. In *ACM Transactions on Graphics (TOG)*, Vol. 30. ACM, 59.

H. Dai, A. Valenzuela, and R. Tedrake. 2014. Whole-body motion planning with centroidal dynamics and full kinematics. In *2014 IEEE-RAS International Conference on Humanoid Robots*. 295–302.

Martin de Lasa, Igor Mordatch, and Aaron Hertzmann. 2010. Feature-based locomotion controllers. *ACM Transactions on Graphics* 29, 4 (2010), 131.

Peter Dorato, Vito Cerone, and Chaouki Abdallah. 1994. *Linear-Quadratic Control: An Introduction*. Simon & Schuster.

Johannes Englsberger, Christian Ott, and Alin Albu-SchÃdffer. 2015. Three-Dimensional Bipedal Walking Control Based on Divergent Component of Motion. *IEEE Transactions on Robotics* 31, 2 (April 2015), 355–368.

Tom Erez, Kendall Lowrey, Yuval Tassa, Vikash Kumar, Svetoslav Kolev, and Emanuel Todorov. 2013. An integrated system for real-time model predictive control of humanoid robots. In *2013 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 292–299.

Farbod Farshidian, Edo Jelavić, Asutosh Satapathy, Markus Giftthaler, and Jonas Buchli. 2017. Real-time motion planning of legged robots: A model predictive control approach. In *2017 IEEE-RAS International Conference on Humanoid Robots*.

Martin L Felis and Katja Mombaur. 2016. Synthesis of full-body 3-D human gait using optimal control methods. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 1560–1566.

Thomas Geijtenbeek and Nicolas Pronost. 2012. Interactive character animation using simulated physics: A state-of-the-art review. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 2492–2515.

Sehoon Ha, Yuting Ye, and C. Karen Liu. 2012. Falling and landing motion control for character animation. *ACM Transactions on Graphics* 31, 6 (2012), 155.

Perttu Hämäläinen, Sebastian Eriksson, Esa Tanskanen, Ville Kyrki, and Jaakko Lehtinen. 2014. Online motion synthesis using sequential monte carlo. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 51.

Perttu Hämäläinen, Joose Rajamäki, and C Karen Liu. 2015. Online control of simulated humanoids using particle belief propagation. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 81.

Daseong Han, Haegwang Eom, Junyong Noh, and Joseph S Shin. 2016. Data-guided model predictive control based on smoothed contact dynamics. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 533–543.

Nikolaus Hansen and Andreas Ostermeier. 1996. Adapting Arbitrary Normal Mutation Distributions in Evolution Strategies: The Covariance Matrix Adaptation. In *International Conference on Evolutionary Computation*. 312–317.

Nicolas Heess, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, Ali Eslami, Martin Riedmiller, et al. 2017. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286* (2017).

Alexander Herzog, Stefan Schaal, and Ludovic Righetti. 2016. Structured contact force optimization for kino-dynamic motion generation. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2703–2710.

Jessica K. Hodgins, Wayne L. Wooten, David C. Brogan, and James F. O'Brien. 1995. Animating human athletics. In *ACM SIGGRAPH*. 71–78.

Daniel Holden, Taku Komura, and Jun Saito. 2017. Phase-functioned Neural Networks for Character Control. *ACM Transactions on Graphics* 36, 4 (2017), 42:1–42:13.

Seokpyo Hong, Daseong Han, Kyungmin Cho, Joseph S Shin, and Junyong Noh. 2019. Physics-based Full-body Soccer Motion Control for Dribbling and Shooting. *ACM Transactions on Graphics* 38, 4 (2019).

Marco Hutter, Christian Gehring, Dominic Jud, Andreas Lauber, C. Dario Bellicoso, Vassilios Tsounis, Jemin Hwangbo, Karen Bodie, Peter Fankhauser, Michael Bloesch, Remo Diethelm, Samuel Bachmann, Amir Melzer, and Mark Hoepflinger. 2016. ANYmal - a highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 38–44.

Jaepyung Hwang, Jongmin Kim, Il Hong Suh, and Taesoo Kwon. 2018. Real-time Locomotion Controller using an Inverted-Pendulum-based Abstract Model. 37, 2 (2018), 287–296.

Satoru Ishigaki, Timothy White, Victor B Zordan, and C Karen Liu. 2009. Performance-based control interface for character animation. In *ACM Transactions on Graphics (TOG)*, Vol. 28. ACM, 61.

Sumit Jain, Yuting Ye, and C Karen Liu. 2009. Optimization-based interactive motion synthesis. *ACM Transactions on Graphics (TOG)* 28, 1 (2009), 10.

Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Kazuhito Yokoi, and Hirohisa Hirukawa. 2003. Biped walking pattern generation by using preview control of zero-moment point. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*. 1620–1626.

Jonas Koenemann, Andrea Del Prete, Yuval Tassa, Emanuel Todorov, Olivier Stasse, Maren Bennewitz, and Nicolas Mansard. 2015. Whole-body model-predictive control applied to the HRP-2 humanoid. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2015)*. 8p.

Taesoo Kwon and Jessica Hodgins. 2010. Control systems for human running using an inverted pendulum model and a reference motion capture sequence. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 129–138.

Taesoo Kwon and Jessica K. Hodgins. 2017. Momentum-Mapped Inverted Pendulum Models for Controlling Dynamic Human Motions. *ACM Transactions on Graphics* 36, 4 (2017).

Yoonsang Lee, Sungeun Kim, and Jehee Lee. 2010. Data-driven biped control. *ACM Transactions on Graphics* 29, 4 (2010), 129.

Sergey Levine and Vladlen Koltun. 2013. Guided policy search. In *International Conference on Machine Learning*. 1–9.

C Karen Liu and Zoran Popović. 2002. Synthesis of complex dynamic character motion from simple animations. *ACM Transactions on Graphics (TOG)* 21, 3 (2002), 408–416.

Libin Liu and Jessica Hodgins. 2018. Learning basketball dribbling skills using trajectory optimization and deep reinforcement learning. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 142.

Libin Liu, Michiel van de Panne, and KangKang Yin. 2016. Guided Learning of Control Graphs for Physics-Based Characters. *ACM Transactions on Graphics* 35, 3 (2016).

Libin Liu, KangKang Yin, and Baining Guo. 2015. Improving Sampling-based Motion Control. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 415–423.

Libin Liu, KangKang Yin, Michiel van de Panne, and Baining Guo. 2012. Terrain runner: control, parameterization, composition, and planning for highly dynamic motions. *ACM Transactions on Graphics* 31, 6 (2012), 154–1.

Libin Liu, KangKang Yin, Michiel van de Panne, Tianjia Shao, and Weiwei Xu. 2010. Sampling-based contact-rich motion control. *ACM Transactions on Graphics* 29, 4 (2010), 128.

Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. 2018. Plan Online, Learn Offline: Efficient Learning and Exploration via Model-Based Control. *arXiv preprint arXiv:1811.01848* (2018).

Adriano Macchietto, Victor Zordan, and Christian R. Shelton. 2009. Momentum control for balance. *ACM Transactions on Graphics* 28, 3 (2009), 1–8.

Igor Mordatch, Martin de Lasa, and Aaron Hertzmann. 2010. Robust physics-based locomotion using low-dimensional planning. *ACM Transactions on Graphics* 29, 4 (2010), 71.

Igor Mordatch, Kendall Lowrey, Galen Andrew, Zoran Popovic, and Emanuel V Todorov. 2015. Interactive control of diverse complex characters with neural networks. In *Advances in Neural Information Processing Systems*. 3132–3140.

Igor Mordatch, Emanuel Todorov, and Zoran Popović. 2012. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 43.

Uldarico Muico, Yongjoon Lee, Jovan Popović, and Zoran Popović. 2009. Contact-aware nonlinear control of dynamic characters. *ACM Transactions on Graphics* 28, 3 (2009), 1–9.

David E Orin, Ambarish Goswami, and Sung-Hee Lee. 2013. Centroidal dynamics of a humanoid robot. *Autonomous Robots* 35, 2-3 (2013), 161–176.

Soohwan Park, Hoseok Ryu, Seyoung Lee, Sunmin Lee, and Jehee Lee. 2019. Learning predict-and-simulate policies from unorganized human motion data. *ACM Transactions on Graphics (TOG)* (2019).

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018. DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills. *ACM Transactions on Graphics (TOG)* 37, 4 (2018).

Xue Bin Peng, Glen Berseth, and Michiel Van de Panne. 2016. Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 81.

Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel Van De Panne. 2017. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 41.

Sarah Pilliner, Samantha Elmhurst, and Zoe Davies. 2009. *The Horse In Motion*. Blackwell Science.

B. Ponton, A. Herzog, S. Schaal, and L. Righetti. 2016. A convex model of humanoid momentum dynamics for multi-contact motion generation. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. 842–849.

Joose Rajamäki and Perttu Hämäläinen. 2017. Augmenting sampling based controllers with machine learning. In *Proceedings of the 2017 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 1–9.

Diana Serra, Camille Brasseur, Alexander Sherikov, Dimitar Dimitrov, and Pierre-Brice Wieber. 2016. A Newton method with always feasible iterates for Nonlinear Model Predictive Control of walking in a multi-contact situation. In *2016 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 932–937.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15 (2014), 1929–1958.

Toru Takenaka, Takashi Matsumoto, and Takahide Yoshiike. 2009. Real time motion generation and control for biped robot -1st report: Walking gait pattern generation-. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 1084–1091.

Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. 2018. Sim-to-Real: Learning Agile Locomotion For Quadruped Robots. *arXiv:1804.10332 [cs]* (May 2018). arXiv: 1804.10332.

Yuval Tassa, Tom Erez, and Emanuel Todorov. 2012. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. 4906–4913.

Steve Tonneau, Pierre Fernbach, Andrea Del Prete, Julien Pettré, and Nicolas Mansard. 2018. 2PAC: Two-Point Attractors for Center Of Mass Trajectories in Multi-Contact Scenarios. *ACM Transactions on Graphics (TOG)* 37, 5 (2018), 176.

Yao-Yang Tsai, Wen-Chieh Lin, Kuangyou B Cheng, Jehee Lee, and Tong-Yee Lee. 2010. Real-time physics-based 3d biped character animation using an inverted pendulum model. *IEEE Trans. Visualization and Computer Graphics* 16, 2 (2010), 325–337.

Pierre-Brice Wieber, Russ Tedrake, and Scott Kuindersma. 2016. Modeling and control of legged robots. In *Springer handbook of robotics*. Springer, 1203–1234.

Alexander W Winkler, C Dario Bellicoso, Marco Hutter, and Jonas Buchli. 2018. Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Robotics and Automation Letters* 3, 3 (2018), 1560–1567.

Zhaoming Xie, Patrick Clary, Jeremy Dao, Pedro Morais, Jonathan Hurst, and Michiel van de Panne. 2019. Iterative Reinforcement Learning Based Design of Dynamic Locomotion Skills for Cassie. *arXiv:1903.09537 [cs]* (March 2019). arXiv: 1903.09537.

Yuting Ye and C Karen Liu. 2010. Optimal feedback control for character animation using an abstract model. *ACM Transactions on Graphics* 29, 4 (2010), 74.

KangKang Yin, Kevin Loken, and Michiel van de Panne. 2007. SIMBICON: simple biped locomotion control. *ACM Transactions on Graphics* 26, 3 (2007), 105.

Wenhao Yu, Greg Turk, and C Karen Liu. 2018. Learning symmetric and low-energy locomotion. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 144.

He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. 2018. Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 145.

Tianhao Zhang, Gregory Kahn, Sergey Levine, and Pieter Abbeel. 2016. Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search. In *2016 IEEE international conference on robotics and automation (ICRA)*. 528–535.

Daniel Zimmermann, Stelian Coros, Yuting Ye, Robert W Sumner, and Markus Gross. 2015. Hierarchical planning and control for complex motor tasks. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 73–81.

APPENDIX

A CHARACTER MODELS

We work with the following four characters models:

Luxo monopod The model has 12 DOFs including six unactuated DOFs at the root link, represented by the head (Figure 13(a)). The neck, knee, and ankle joints have 3, 1, and 2 DOFs, respectively. It has a mass of 40.6 kg, two thirds of which is concentrated in the head.

Humanoid biped The model has 32 DOFs; knees and elbows have one DOF, the ankles have 2 DOFs, and the remaining joints have 3 DOFs (Figure 13(b)). It has a mass of 69.8 kg.

Cassie biped robot The model has six DOFs for each limb, a total of 18 DOFs, and a mass of 31 kg (Figure 13(c)).

ANYmal quadruped robot The model has three DOFs for each limb, a total of 18 DOFs, and a mass of 60 kg [Hutter et al. 2016]. We also use a large variant of the ANYmal model to reproduce gaits common to horses. The large ANYmal model has a mass of 300 kg.

For the Cassie model, the mass and inertia matrix are obtained from an open-source specification. For all the other models, the mass and inertia matrix of each body part are computed based on a uniform density assumption and the total mass of the each model.

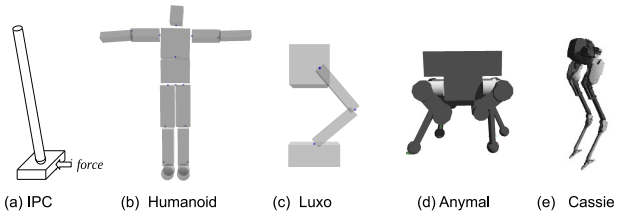


Fig. 13. The character models used with our system.

B PENDULUM TRAJECTORY PLANNER DETAILS

We use an infinite-horizon linear quadratic regulator (LQR) for controlling the IPC to follow the user-defined desired direction and speed because it is highly efficient and stable enough for our purposes [Dorato et al. 1994]. The desired speed for the pendulum is scaled down by $cos\,(\theta)$ from the user-specified value, when the average slope of the terrain θ in the planning horizon is non-zero. The average slope is computed from a plane fitted using uniformly sampled points along the cart trajectory offset by 20 cm both laterally and sagittally. For the plane fitting, we use a Bayesian linear regression scheme to avoid overfitting when terrain is rough [Box and Tiao 1992]. The modified desired speed is used as the target speed of the LQR controller so that the resulting cart trajectory on the terrain closely matches the user-specified desired speed. Note that the dynamics of the horizontal position and the leaning angle of the IPC is independent of the terrain slope when the mass of the cart is zero because it is governed by horizontal force only and the vertical dynamics is fully constrained by the terrain (as is typical for locomotion planning). Therefore the rough COM plan vertically projected onto the terrain is relevant for level and sloped surfaces.

The inverted pendulum model by itself cannot represent the facing direction of the character. Instead, we define the facing direction using a Hermite spline that spans a fixed-duration future time-horizon, which is long enough to contain 1–2 footsteps per limb for all limbs. Together with the two rotational joint angles, this facing direction constructs the orientation of the pendulum (ϕ). When the user specifies a new desired direction, this spline is re-created to smoothly interpolate the current and desired orientation. The user-defined desired speed is converted into the desired velocity for the pendulum model based on the orientation obtained from the spline with the modified magnitude in the forementioned manner. In this way, the desired velocity is always defined relative to the current orientation.

C FOOTSTEP PLANNER DETAILS

C.1 Foot Offset Transformation

In the footstep planner, the actual footstep location is determined by transforming the sample location by the user-defined foot offset transformation. which is composed of the lateral translation and the rotation about a vertical axis from the cart to the foot (Figure 14).

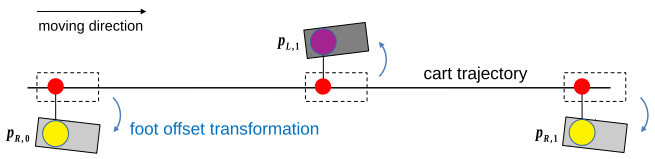


Fig. 14. An example of the cart trajectory and footstep locations in straight walking of the humanoid character. Each sample location (red circles) is transformed to the actual footstep location (yellow and purple circles, denoted by $p_{i,j}$), which corresponds to the heel in this example, by the user-define foot offset transformation. To demonstrate the effect of its rotational component, we draw the shape of actual feet before (dotted boxes) and after (gray-shaded boxes) the foot offset transformation.

C.2 Optimization Variables Details

Limb index i. Each limb can be associated with multiple end-effectors, e.g., the heel and toe for the human model. However, such end-effectors share the same set of variables when planning a motion sketch, and thus do not need to be considered in the footstep planner.

Footstep index per limb j. Note that j can be equal to or less than zero to describe the footstep whose timing and location are already determined from the past planning. Specifically, $\{t_{i,0}\}_{\forall i}$ are fixed constants which are already determined during the last planning.

Number of footsteps of i-th limb in the horizon $n^i_{contact}$. In practice, n^i_{contact} is set such that more contacts than n^i_{contact} usually occur during the current planning horizon. Thus the sampling time $t_{i,j}$ is usually within the horizon. Occasionally when $t_{i,j}$ is sampled outside the planning horizon, we linearly extrapolate the planned pendulum trajectory to determine the corresponding footstep location.

C.3 Constraints Details

Constraint for temporal order in a single limb. Note that $t_{i,0}$ is not an optimization variable as it indicates the fixed footstep timing determined by the previous footstep planning.

Constraint for temporal order across multiple limbs. For example, the constraint $t_{L,j+1} - t_{R,j} > 0$ and $t_{R,j} - t_{L,j} > 0$ are used in order to allow the left and right legs to alternate in the walking motion. Multiple types of quadruped gaits can also be generated using a single set of constraints as described in Appendix G.

Constraint for obstacle avoidance. Intersection tests are performed with a capsule shape that approximates the conctact geometry of a foot. For humanoid models, the capsule shape is constructed so that the heel and toe coincide with the two centers of the capsule.

C.4 Constraints for Derivative-Free Optimization

The derivative-free optimization supports directly implementing hard-constraints by guaranteeing that all the random samples satisfy the constraints. Hard constraints are used for controling the the temporal order of the footsteps and for preventing the footsteps from colliding with obstacles (Section 4.2.3). Specifically, $t_{i,j+1}$ is encoded relative to $t_{i,j}$, and the difference in time is guaranteed to be sampled in a range greater than zero. Similarly, when footstep position $\mathbf{p}_{i,j}$ corresponding to sampling time $t_{i,j}$ intersects with any obstacle, we simply discard the sample, and then sample again. if collisions remain even after a predetermined number of resampling attempts, we use a time-based line-search that moves colliding footsteps towards the nearest non-penetrating location. The optimized timing uses the best sample observed during the entire optimization process, so is guaranteed collision-free.

C.5 Footstep Contact Duration Details

We heuristically adjust the contact duration of each limb based on the forward speed so that the COM moves a constant distance during the support phase. That is, as the speed increases, the contact duration decreases and the swing phase duration increases because the temporal difference between two successive sampling times are fixed by the optimization. As a result, it triggers a running motion. And if the speed is lowered, the support phase will have a longer duration as observed in walking. The user-specified reference contact duration at specfic moving speed is used to calculate the constant moving distance of COM. For example, if a user specifies the reference contact duration of 0.2 s at speed of 4 m/s, the COM moving distance is $0.2 \times 4 = 0.8$ m, thus the actual contact duration is 0.4 s when the character's speed decreases to 2 m/s. Note that each sample timing is used as the midpoint of the contact duration of the corresponding limb (Figure 5). If a single limb has multiple end-effectors, such as a heel and toe of a humanoid leg, it is used as the midpoint of the overlapping duration among the contact duration of these end-effectors. The ratio for end-effector contact phases, such as heel only, heel and toe, and toe only phases for a humanoid, can be specified by the user, or easily can be extracted from a reference motion if supplied.

C.6 Stepping Stones Problems

The Section 4 describe the "standard" process of the motion sketch generator used in ordinary cases, in which the pendulum trajectory is first planned and then the footsteps are planned with the output of the first step, as depicted in Figure 2. However, the planning order of pendulum trajectory and footstep can be reversed for some cases thanks to the flexibility of our system.

In some of our results, the biped character must step on irregularly-placed stones (Figure 10(f)) or on regularly-spaced stairs (Figure 10(g)) in front of the character. In these stepping stones cases, the character first needs to decide which stones to step on, and then the pendulum trajectory can be created to follow determined stone locations. We formulate this type of problems using a similar optimization scheme but using the sequence of stones instead of the sampling times as independent variables. This formulation is different from the ordinary one in that the footstep location should be determined before the cart trajectory, and the optimization variables are now discrete variables. We use a standard genetic algorithm to solve this optimization problem. To solve the stepping stones problems, we use a standard genetic algorithm instead of CMAes because the optimization variables are discrete. A genetic algorithm statistically selects fittest chromosomes from the current population, and uses them to produce the offspring by repetitive crossovers and random mutations. An integer array representing a sequence of stones is used as a chromosome in our formulation. We use two different chromosome encoding schemes depending on the stepping stones type.

In the first scheme, a Boolean array representing a sequence of three candidate future stones is used as a chromosome. This scheme is used for sequential stepping stone scenarios, Humanoid-StepUpDown and Humanoid-StairWalk. Specifically, we assume that each stone can be stepped 0, 1, or 2 times. For example, some stones will be stepped on by both feet, and some stones will be stepped on by one foot or not. A sequence of stones can then be represented as a Boolean array where two consecutive bits represents a stone. The number of true bits denotes the number of times the stone is stepped on. A single-stepped stone is more likely to appear in mutation and crossover operations than zero or two stepped one because it is represented by both 01 and 10 bit sequences, which is designed on purpose for natural animation.

The second scheme is for more complex stepping-stones environments. This scheme is used for randomly scattered stepping stone scenarios, Humanoid-Stones and Humanoid-TerrainStones. An integer array containing the index of stones for future three footsteps is used as a chromosome. In these scenarios, the stones can be stepped on in an arbitrary order, and thus the search space is much larger compared to the sequential stones scenario given the same number of stones. Duplicate numbers in a chromosome are not allowed to prevent the character from stepping on the same stone twice. Half of the population uses random scrambles for mutation, instead of a single point mutation.

From a chromosome, footstep positions $\{\mathbf{p}_{i,j}\}_{\forall i,j}$ can be decoded. A set of sampling times $\{t_{i,j}\}_{\forall i,j}$ can also be computed using a constant speed assumption. Therefore, the same objective functions can be used for optimization. We empirically choose the mutation

probability of 0.015, crossover probability of 0.5, population size of 30, and maximum iteration limit of 300. Once, the sequence of stones for a future horizon is found, we generate a corresponding pendulum trajectory that closely follows a piece-wise linear spline interpolating the stone locations using a space-time optimization scheme in [Kwon and Hodgins 2017].

D CDM TRAJECTORY PLANNER DETAILS

In centroidal dynamics, an articulated model can be simplified as a single rigid body, which represents the CDM and has the same mass, COM and overall inertia , which is called centroidal inertia [Orin et al. 2013] (Figure 7). The orientation of the CDM can be interpreted as the overall orientation of the articulated model. Any external force including ground contact force can be applied to the CDM at any point to change the acceleration of the body. Similar to the IPC in Section 4.1, the CDM for each character is constructed with the COM and inertia calculated from the rest pose or the average pose of the reference motion if supplied.

D.1 Differences from [Winkler et al. 2018]

Our work differs from [Winkler et al. 2018] in multiple respects: (a) We solve to optimize an objective function that makes CDM trajectory follow the motion sketch trajectories, in addition to satisfying the hard constraints. (b) We employ a derivative constraint on the ground contact forces, which helps prevent excursions of the contact-force spline outside of the friction cones, as occurs on occasion in [Winkler et al. 2018]. (c) We use multiple planes to enforce leg-length constraints, as we found the use of body-oriented box constraints to be problematic. (d) Our formulation allows to use multiple end-effectors for each limb to generate natual locomotion behavior, such as foot rolling in human locomotion, which is represented by separate heel-and-toe contacts. (e) Our CDM optimization is effectively combined with the IPC-based plans, which allows: (e1) automated physically-feasible footstep generation by user-defined control objectives. (e2) improved robustness of CDM optimization by using the IPC solution as initial guess. (e3) shorter CDM planning horizon which speeds up solving the CDM optimization, which is the benefit of the improved robustness.

D.2 Optimization Variables Details

In this planner, the COM position and orientation, and contact location and force are calculated through an optimization process. The CDM motion and contact forces are represented as cubic Hermite splines. Note that the contact locations, which have already solved in the footstep planner, are solved again during CDM trajectory optimization. If both the contact timing and the positions are fixed, the CDM motion has to be generated only by the change of the contact force, which seriously restricts the robustness of the NLP solver. We allow the contact positions to move away from the planned position for the improved robustness of NLP solver and higher quality of resulting motion.

CDM motion. The sequence of splines for the CDM trajectory (COM position and orientation) interpolates $\mathbf{q}$ and $\dot{\mathbf{q}}$, which are a part of the optimization variables, as uniformly distributed keys. $\mathbf{q} \in \mathbb{R}^{6n}$ is the concatenated vector of the positions and orientations of the CDM at keyframes and $\dot{\mathbf{q}} \in \mathbb{R}^{6n}$ is its time derivative, where n is the number of keyframes in T. Note that two consecutive groups of six numbers in $\mathbf{q}$ and $\dot{\mathbf{q}}$ defines a single Hermite spline segment. The orientation is parameterized by Euler angles. For each planning step, the positions and Euler angles $\mathbf{q}$ are represented relative to the initial state (position and orientation) of the CDM to avoid the possible singularity of the Euler angles. The keyframes can be thought of as simulation timesteps, as the equations of motion is enforced at each keyframe, similar to the simulation timesteps in typical forward dynamics simulation. The smaller both the simulation timestep, the more accurate simulation but more time for computing. We empirically set the temporal difference between keyframes by choosing the number of keyframes to generate high-quality motion with reasonable speed. We experimentally verified that using more splines than this level do not visibly change the results.

Contact forces. For the contact forces, a sequence of four spline segments $f_{l,j}(t)$ is used for each contact duration at j-th footstep of each end-effector l ($l \in [1, n_{\text{effector}}], j \in [0, n^l_{\text{eff-contact}}]$ where n_{effector} is the total number of end-effectors across all limbs and $n^l_{\text{eff-contact}}$ represents the number of footsteps for l-th effector in T). The contact force is defined for each end-effector to generate natual locomotion behavior. Note that j starts from zero which means that the contact force of the already determined 0-th footstep is optimized as well because $f_{l,0}(t)$ is the important means of affecting the CDM trajectory until the 1-th footstep, contrary to the footstep planner which does not use the 0-th footstep as an optimization variable. For each planning, the $f_{l,0}(t)$ is initialized to the splines determined from the last planning to provide better initial guess. Each spline segment is defined by two sets of $\mathbf{f}_{l,j}, \dot{\mathbf{f}}_{l,j} \in \mathbb{R}^3$, which are end-point force and force derivative conditions, and adjacent segments share them at every connection point. The concatenated vectors $\mathbf{f}, \dot{\mathbf{f}} \in \mathbb{R}^{5\cdot3\cdot n^{\text{total}}_{\text{eff-contact}}}$ are a part of the optimization variables, where $n^{\text{total}}_{\text{eff-contact}}$ is the total number of end-effector contacts in T including the 0-th footstep ($n^{\text{total}}_{\text{eff-contact}} = \sum_{l=1}^{n_{\text{effector}}} n^l_{\text{eff-contact}} + 1$).

Contact positions. Each contact position $\mathbf{p}_{i,j}$ ($i \in [1, n_{\text{limb}}], j \in [0, n^i_{\text{contact}}]$) is concatenated to a single vector $\mathbf{p} \in \mathbb{R}^{3\cdot n^{\text{total}}_{\text{contact}}}$ to be a part of the optimization variables, where $n^{\text{total}}_{\text{contact}}$ is the total number of limb contacts in T including the 0-th footstep ($n^{\text{total}}_{\text{contact}} = \sum_{i=1}^{n_{\text{limb}}} n^i_{\text{contact}} + 1$). Each contact position is accessed by limb index i instead of an effector index because multiple end-effectors in the same limb are relatively fixed to each other. In constrast to the contact force, the already determined position $\mathbf{p}_{i,0}$ should not change to avoid foot-skate artifacts. Instead of excluding $\mathbf{p}_{i,0}$ from the set of optimization variables, we use an equality constraint to exactly locate it at the last planned value, which simplifies implementation. For later contact positions, the values from the motion sketch are used as initial guesses.

D.3 Constraints Details

Equation of motion. Although the actual inertia matrix of a character is state-dependent, the CDM assumes that the inertia of the character is independent from the actual joint configuration. This

assumption is an approximation when the joint angles deviates from the reference motion, but it allows the formulation to be simple and enables a fast solution.

Constraints for contact positions. In some cases, such as the stepping stones or monkey bars experiments, the character needs to keep its contact locations close to the specified locations. We formulate this with the optional constraint Equation 17 that sets bounds on contact positions $\mathbf{p}_{i,j}$ when they need to stay close to the specified locations $\mathbf{p}^0_{i,j}$ in a certain threshold ϵ_p, which is set to 6 cm in our experiments. The specified locations $\mathbf{p}^0_{i,j}$ might be planned locations from the motion sketch or a user-specified contact plan. Equation 17 is not used for other ordinary cases, but nevertheless, the CDM optimization do not yield the contact positions far from the one in the motion sketch because of using them as the initial guess, using the trajectories in the motion sketch as the desired trajectory, fixed contact timings, and the leg-length constraint (Equation 14).

D.4 Desired CDM trajectory in Objective Function

The desired CDM trajectory $\bar{\mathbf{q}}$ used in the objective function (Equation 20) is defined by sum of the pendulum trajectory and the user-defined horizontal oscillatory displacement of the COM.

Pendulum trajectory. The pendulum trajectory is a 6-DOF trajectory combining the COM position and pendulum orientation trajectories ($\mathbf{p}_\text{pend}$, $\mathbf{R}_\text{pend}$) from the motion sketch. To provide smooth vertical positions for the COM considering the terrain, the pendulum is assumed to move on a plane fitted using the height samples near the cart trajectory, as described in Appendix B.

Horizontal oscillatory displacement. The vertical oscillation of the CDM is naturally generated by the CDM optimization under the given contact conditions. Its horizontal oscillation, however, is not generated only provided with the smooth COM trajectory from the motion sketch because CDM optimization moves the contact position in a line (for ordinary cases not using Equation 17) to create the CDM trajectory that exactly follows the desired COM trajectory without any oscillation. By specifying the horizontal oscillatory displacement, a user can adjust the degree of the horizontal oscillation of the character locomotion.

E MOMENTUM-MAPPED INVERSE KINEMATICS DETAILS

The momentum-mapped inverse kinematics takes as input the resulting pose of the previous frame as well as the reference pose to consider both joint angles and velocities. Each frame of an optional reference motion can be used as the reference pose to give the stylistic details to the character's motion, if supplied, otherwise the rest pose of the character model is used as the reference pose. For example, the motions of the Luxo monoped, the Cassie biped, and the ANYmal quadruped are generated without using any reference motion. If a reference motion is supplied, each behavior can be successfully generated using a single short reference motion clip containing a single gait cycle for each limb. That is, two cycles are sufficiently long for any type of locomotion.

Both the input and the output poses of the momentum-mapped inverse kinematics solver use the same representation that uses angles instead of quaternions for all degrees of freedom (DOFs) for efficiency. The rotational DOFs of the root joint are represented using Euler angles appropriately chosen to avoid the possible singularity.

Regarding the first term of the cost function, $0 \leq i_k \leq 1$ is the weighting factor for the k-th end-effector and is set 1 for most cases. The only exception is the case when a flight phase is long, for example, leaps, jumps and flips. In such cases, i_k makes continuous transition from one to zero when the end-effector is far from the previous support position than a threshold. Conversely, it makes transition from zero to one as the end-effector approaches the planned landing position. The only exception is the case when a flight phase is long, for example, leaps, jumps and flips, where the end-effector is far from the previous support position than a threshold. In such cases, i_k makes continuous transition from one to zero using a parabola centered at the middle of the swing phase, and back to one as the end-effector approaches the planned landing position. $\overline{\mathbf{y}}_k$ is set to the planned footstep location for stance phase, and to the swing foot trajectory computed by the arc interpolation of the footstep locations [Hwang et al. 2018] for swing phase.

For the second term, the generalized position of the planned CDM can be estimated from the desired pose $\overline{\mathbf{x}}$ because the desired pose is transformed so that its CDM matches the planned CDM. Another way to interpret this term is finding $\mathbf{x}$ so that moving from $\mathbf{x}$ to $\overline{\mathbf{x}}$ does not generate any velocity in terms of centroidal dynamics, meaning that CDM configurations of these two poses should coincide with each other. Both $\mathbf{I}_\text{F}$ and $\mathbf{J}$ are obtained using the previous pose for computational efficiency so as not to reevaluate them at every evaluation of objective and gradient functions during the optimization process.

When using a reference motion, we annotate the contact state independently for each limb. For each limb, a Boolean contact state is assigned for each frame. Based on this annotation, a locomotion phase $\psi (0 \leq \psi \leq 1)$ is automatically assigned using a linear function for each limb. The midpoint of a contact duration of a limb corresponds to $\psi = 0$, and the midpoint of the next contact duration of the limb corresponds to $\psi = 1$, which is equivalent to $\psi = 0$, and ψ increases from 0 to 1 for the time in between. A set of phase values $\{\psi_i\}, i \in \{1, 2, \cdots, n_\text{limb}\}$ is used to retrieve a reference pose corresponding to the phase values when generating the final motion.

F LEARNING FRAMEWORK DETAILS

F.1 Network Input Details

Initial model state. The initial CDM state, feet positions, and the root position and orientation of the full-body pose are encoded locally to the pendulum frame, which is constructed from the COM position and pendulum orientation of the IPC, at the initial time of the CDM planning horizon. The remaining joint angles of the full-body pose are represented with respect to their parent joint frames.

Future motion information. At each sample time, we extract a number of features such as the pendulum configuration, phases $\{\psi_i\}$ for all limbs, and the heights of the terrain at four locations 20 cm away to the left, right, front, and back of the cart position of

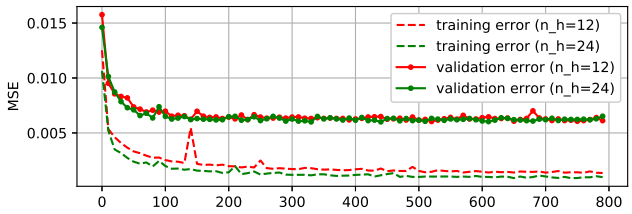


Fig. 15. The mean squared error (MSE) of trained and validation data in terms of epoches for two different hyperparameter choices ($n_h = 12$ and $n_h = 24$).

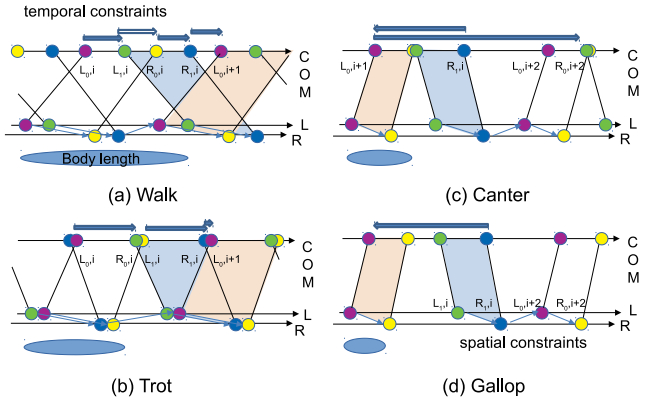


Fig. 16. The relative position between the center of mass and a foot of a horse observed in the four different gait styles of the horse (walk, trot, canter, and gallop). The purple and yellow dots represent the rear legs, and the green and blue dots represent the front legs. The blue ellipsoid represents the body length of the horse relative to the stride length.

IPC. All the heights are represented relative to the average height of all locations around sample cart positions for the planning horizon, which is proven to significantly improve prediction accuracy and sample efficiency of the network training. Each phase $\psi_i \in [0, 1]$ for limb i is stored as two values, $cos(\psi_i \cdot 2\pi)$ and $sin(\psi_i \cdot 2\pi)$ to avoid discontinuity. If the output full-body motion is generated using multiple reference motions, the optional information motion type is recorded with the motion sketch and extracted at each sample time.

F.2 Network Output Details

The pendulum is assumed to move on a plane fitted using the height samples near the cart trajectory as described in Appendix B, which prevents jerky trajectories from being produced with the pendulum frame. Possible alternative approaches are using a pendulum projected to a horizontal plane which has the same average height, or using a pendulum moving exactly on the terrain. The former does not generalize well to sparse situations with fewer examples even though the input for the network contains the height information. The latter is undesirable in that the smooth CDM states are encoded relative to a jerky trjaectory when the terrain contains rough geometry. The plane relative encoding scheme produces overall satisfactory results.

The network output consists of uniformly sampled frames over the planning horizon (experimentally set as 30). The dimension of each sample frame depends on the character model. For example, the output dimension for Humanoid scenarios is 1500 (= 30 × 50).

Figure 15 shows the trends of trained and validation data in terms of epoch versus the mean squared error (MSE) for two cases ($n_h = 12$ and $n_h = 24$ in Section 7.2). The validation set contains about 6% of the total dataset. As shown in the figure, the use of more number of hidden neurons does not improve the accuracy over the validation dataset, though the accuracy over the training dataset increases. This is a clear sign of overfitting. Similarly, the accuracy over the validataion stays the same after about 400 epoches. We used such observations when choosing hyperparameters because an overfitted network does not generalize well to unobserved situations.

G QUADRUPED GAITS

Quadruped gaits are unique in that the temporal order in which footsteps occur depends on the type of gait, e.g., walk, trot, pace, canter, and gallop. Preparing a reference motion and explicitly modeling the phase structure for each type of the gaits can be a time-consuming task, especially when the transitions between the gait types also need to be considered. However, a principal reason that many gait patterns are difficult understand is that the conventional view depicts the footstep patterns on the time axis. In contrast, if we plot the spatial locations of the footsteps observed from the animated pictures of a typical horse along the forward facing direction [Pilliner et al. 2009], we can observe that there is an interesting similarity between these seemingly completely different gaits. Specifically, as shown in Figure 16d, the footsteps (green, blue, purple, and yellow dots) appear in the same order across the gait types in the spatial axis. The blue quads in Figure 16 represent front legs and the orange quads the rear legs. As the running speed increases, the COM part of the orange quad moves continuously to the left of that of the blue quad in the space-time graph. We configure the constraints and objectives of the footstep planner based on this observation, and experimentally show that the four kinds of gaits can emerge from a single objective function.

The movement of the quads is a consistent pattern observed from videos and pictures of actual horses. The purple dot $L_{0,i+1}$ shown in Figure 16a moves beyond the green dot $L_{1,i}$ in Figure 16d. Therefore, the same blue arrow connecting $L_{0,i+1}$ and $L_{1,i}$ points to the right in Figure 16a and to the left in Figure 16d. Intuitively, when the horse gallops quickly, the body length of the horse is relatively short compared to the stride length, so the distance between the front legs and the rear legs has less impact on the temporal order, and the temporal order and spatial order of the footsteps coincide. However, as the speed decreases, the stride of the horse becomes shorter and the temporal order begins to change. In other words, the facts that these gaits are observed at different velocities and that there is a large positional offset between the front legs and the rear legs causes the unique and complex gaits of quadrupeds.

Based on this observation, we first configure the limb temporal order constraints (Equation 8) assuming that all four legs are always used:

$$t_{L_0,i} < t_{L_1,i}, t_{R_0,i}, t_{R_1,i} < t_{L_0,i+2}, \tag{25}$$

where L_0 and R_0 are the rear legs, and L_1 and R_1 are the front legs. Because we do not constraint the order between $t_{L_1,i}$,$t_{R_0,i}$, and $t_{R_1,i}$, all combinations of temporal orders can still be expressed.

Next, we configure the step duration objective (Equation 2) by assuming that the right front leg is the leading leg of the quadruped, which means that the quadruped prefer to place the right front leg in front of the left front leg (Figure 16a):

$$P_{\text{dur}} = \{(L_0, L_1, 0, \delta), (R_0, R_1, 0, \delta), (R_1, L_0, 1, \delta), (L_1, R_0, 0, \delta)\}, \tag{26}$$

where the first two terms are between front and rear legs, and the last two terms are between left and right legs. The last term depicted as white arrow in Figure 16a is used only when the constraint value is negative to allow for asymmetric behavior at faster speeds.

The step length objective (Equation 4) are set based on the observation shown in Figure 16d.

$$P_{\text{stride}} = \{(L_0, R_0, 0, d), (L_1, R_1, 0, d), (R_1, L_0, 2, d)\}. \tag{27}$$

Because the spatial order is maintained regardless of the gait type, we use much higher weight w_4 for the step length objective compared to w_2 for the step duration objective in the footstep planner objective function E_{step} (Equation **??**), and control the overall stride duration using the weight w_1 for the stride duration (Equation 1). The temporal objective between different legs (Equation 8) have lower weights ($w_2 = 1.0$) than the weights ($w_1 = 2.0$) for the stride duration objective (Equation 1). The step length objective (Equation 4) have much higher weights ($w_4 = 40$) to cause the unconstrained gap between the yellow dot ($R_{0,i+1}$) and the green dot ($L_{1,i}$) in Figure 10d, which corresponds to longer flight phases during galloping.

We note that these terms are developed based on our observations, and it is possible that other combinations may produce similar results. Also the same set of constraints can produce different behaviors depending on the body length. For example, the smaller ANYmal character uses pacing instead of galloping at faster speeds because of its shorter body even though we use the same set of constraints for both the ANYmal characters. The process of adjusting the weight of each term and adding more terms when necessary is intuitive and predictable, and thus desirable results can be obtained in a reasonable time, particularly given that motions are synthesized at interactive rates.

H EXPERIMENTAL PARAMETERS DETAILS

For the Luxo, Cassie, and ANYmal models, the final full-body motions are generate only using the rest pose of those models, without any reference motion, and the mass and inertia of the IPC and CDM are calculated from the rest pose as well. For the Humanoid model, a two-second-long walking motion, a 1.5-second-long running motion, and a single forward jumping motion are used for generating all the human behaviors, except a few *Locomotion style variations* scenarios where a reference motion is used to generate each style. The IPC and CDM of the Humanoid model for each experiment are generated from the average pose of the corresponding reference motions. The gait parameters, which are input to the footstep planner, are extracted from the reference motion for the Humanoid model and are pre-defined by the user for other models. They are kept constant during each scenario except for a few *Locomotion style variations* scenarios where some parameters continuously change to generate various styles.

The number of end-effectors in a single limb used in the CDM planning step can be different for each character model. The Luxo, Humanoid, and Cassie models have two end-effectors for each limb and the ANYmal model has one end-effector for each limb.

Table 3. Comparison of our per-segment deep neural network (DNN) and a per-frame phase-functioned neural network (PFNN) for the same dataset (Humanoid). Prediction time is normalized so that the output corresponds to one second. The number of weight parameters of each network is tuned to achieve the best overall result while avoiding overfitting.

Humanoid	Per-segment DNN	Per-frame PFNN
# of data pairs	2722	38227
# of epoches	500	500
training time	4 m 4 s	91 m 38 s
prediction time	2.9 ms	112 ms

I MORE RESULTS

Comparison of different neural network structures. Our trained network outputs the full-body motion segment for the CDM planning horizon at once. We compared our network with conventional per-frame networks (both phase-functioned [Holden et al. 2017] and not). As the phase-functioned per-frame network outperforms the non-phase-functioned one, we report the comparison result only with the phase-functioned network. As shown in the accompanying video and Table 3, our per-segment network works much better with more accuracy, less foot-sliding, and much faster training and prediction speed. This is probably because our dataset is sparse while the character can be controlled in a very wide range of speed, turning speed, and terrain variation.

Validation of the pendulum trajectory and footstep planners. Our CDM-based motion generation system contains three-level planners. We conduct the ablation experiments to identify the effect of each planner. Because the CDM planner is the fundamental component that guarantees the physical correctness of the generated motion, the experiments are carried out by bypassing the two higher-level planners.

Without the pendulum trajectory planner, the CDM trajectory planner fails to find a feasible solution for a simple slow-running Humanoid motion. For this experiment, the COM position trajectory generated by the pendulum trajectory planner is only input for the footstep and CDM trajectory planner. The cart position of the IPC is generated by projecting the COM position to the ground, and the footstep planner generates footsteps based on this projected trajectory. The result shows that the motion sketch planned by the pendulum trajectory is an effective initial guess for the CDM trajectory planner, which significantly improves robustness of the CDM optimization.

In another experiment, we generate Humanoid-StepUpDown motions with and without the footstep planner. Without the footstep

Table 4. Timing statistics for all scenarios. T_{clip} is the length of the generated full-body motion clip for each scenario. T_{comp} is the computation time to generate the clip. $T_{\text{comp}}/T_{\text{clip}}$ indicates the computation time for generating one second full-body motion. Note that T_{comp} for *Learning framework* scenarios only measures the online querying time from the learned neural network. The detailed statistics for *Learning framework* scenarios including data generation and training are given in Table 1.

scenario	T_{clip}	T_{comp}	$T_{\text{comp}}/T_{\text{clip}}$
Walking and running motions			
Humanoid-WalkAndStop	4.2s	2.23s	0.53
Humanoid-SharpTurn	17.5s	8.9s	0.51
Humanoid-RunVaryingSpeed	14.7s	17.35s	1.18
Humanoid-TerrainRun	10.0s	24.4s	2.44
Luxo-Terrain	17.2s	34.7s	2.02
ANYmal-Terrain	12.8s	6m 17s	15.43
Learning framework			
Humanoid-DNN	27.3s	10.37s	0.38
ANYmal-DNN	16.28s	15.62s	0.96
ANYmal-DNNPush	8.3s	8.43s	1.02
Environmental variations			
Humanoid-Leap	12.2s	3m 4s	15.1
Humanoid-TerrainLeap	14.1s	1m 41s	7.17
Humanoid-Hurdles	35.8s	9m 15s	15.5
Humanoid-StepUpDown	15.0s	34.6s	2.31
Humanoid-StairWalk	14.6s	50.1s	3.42
Humanoid-Stones	7.3s	26.8s	3.67
Humanoid-TerrainStones	11.5s	38.9s	3.38
Humanoid-Moon	43.1s	9m 2s	12.57
Locomotion style variations			
Humanoid-VaryingStride	37.1s	41.5s	1.12
Humanoid-VaryingLatFoot	19.5s	11.8s	0.6
Humanoid-ArmWideOpen	23.8s	18.9s	0.79
Humanoid-LeanedBack	23.8s	20.0s	0.84
Humanoid-Sneaky	23.8s	26.9s	1.13
Humanoid-March	23.8s	20.4s	0.85
Humanoid-Spin	8.5s	5.9s	0.69
ANYmal-Gaits	62.4s	18m 24s	17.9
ANYmal-Rush	70.3s	23m 8s	19.7
Cassie-Style1	27.1s	53.4s	1.97
Cassie-Style2	25.6s	53.5s	2.09
Cassie-Style3	15.3s	48.6s	3.18
External pushes			
Humanoid-Push	25.8s	8.8s	0.34
Monkeybars, jumps, and superhuman jumps			
Humanoid-Monkeybars	10.7s	48.9s	4.58
Humanoid-RandomJumps	19.4s	9.9s	0.5
Humanoid-SuperJumps	21.6s	17.0s	0.79
Humanoid-SuperFlips	18.0s	14.4s	0.8

Table 5. Compilation of video links for related work on online and offline trajectory optimization

[Lowrey et al. 2018] https://sites.google.com/view/polo-mpc
[Hämäläinen et al. 2015] https://www.youtube.com/watch?v=awa9MrIbbI0
[Rajamäki and Hämäläinen 2017] https://www.youtube.com/watch?v=1Vcr08Xbtc8
[Mordatch et al. 2012] https://www.youtube.com/watch?v=mhr_jtQrhVA
[Al Borno et al. 2013] https://www.youtube.com/watch?v=dsxbKSG7_bs
[Liu et al. 2010] https://www.youtube.com/watch?v=U7caeI5A7rU
[Tassa et al. 2012] https://www.youtube.com/watch?v=2FJtJDYCQIE
[Kwon and Hodgins 2017] https://youtu.be/uv3YYG4tvsM
[Winkler et al. 2018] https://www.youtube.com/watch?v=0jE46GqzxMM
[Farshidian et al. 2017] https://www.youtube.com/watch?v=EYGVmcd9uds
[Dai et al. 2014] https://www.youtube.com/watch?v=2Vry-th8g2s

planner, the Humanoid models exhibits unnatural and unstable walking motion nearly falling under the stones.

Effect of the velocity terms in the momentum-mapped inverse kinematics. We introduce the velocity terms (the third and last terms in Equation 23) in the momentum-mapped inverse kinematics solver to consider CDM and joint velocities as well as their positions when solving inverse kinematics. To show the effect of these terms, we compare the full-body motion of sharp-turning from our solver and from the solver excluding these two terms (conventional solver), with the same CDM plan input. As shown in the accompanying video, the generated motion from our solver naturally rotates body direction when changing walking direction, while the conventional solver starts changing body direction late so it is changed abruptly compared to our solver.

Real-time demos without using the learning framework. Even without the learned network, our system can generate live-demos for the scenarios that are not too complicated. We present two live-demo examples using our motion generation algorithm alone (Luxo-Live and Humanoid-Live in the supplementary video). For these live demos, we use a multi-threaded implementation where one thread is assigned for the motion generation, and another thread is used for the rendering and the UI. When the rendering thread starts the CDM forward dynamics simulation of the last planned motion (which is a half-cycle long for the biped character), the other thread starts the planning of the next motion based on the current goal position specified by the user. The planning usually finishes before the end of playback of the last planned motion, but occasional motion pauses will occur when the planning takes more time.