Deep Compliant Control

Seunghwan Lee
Department of Computer Science and Engineering
Seoul National University
Seoul, South Korea
lsw9021@gmail.com

Phil Sik Chang
Department of Computer Science and Engineering
Seoul National University
Seoul, South Korea
lasagna1299@gmail.com

Jehee Lee
NCsoft
Kyungkido, South Korea
Seoul National University
Department of Computer Science and Engineering
Seoul, South Korea
jeheel@gmail.com

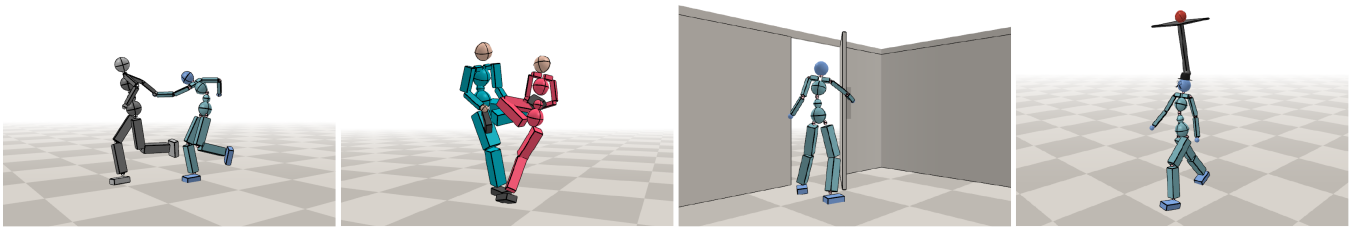

Figure 1: Our compliant controller learns dynamic interactions with environments. (Left to right) Hand-in-hand running, chicken hopping game, opening a door, and balancing a ball.

ABSTRACT

In many physical interactions such as opening doors and playing sports, humans act compliantly to move in various ways to avoid large impacts or to manipulate objects. This paper aims to build a framework for simulation and control of humanoids that creates physically compliant interactions with surroundings. We can generate a broad spectrum of movements ranging from passive reactions to external physical perturbations, to active manipulations with clear intentions. Technical challenges include defining compliance, reproducing physically reliable movements, and robustly controlling under-actuated dynamical systems. The key technical contribution is a two-level control architecture based on deep reinforcement learning that imitates human movements while adjusting their bodies to external perturbations. The controller minimizes the interaction forces and the control torques for imitation, and we demonstrate the effectiveness of the controller with various motor skills including opening doors, balancing a ball, and running hand in hand.

CCS CONCEPTS

• **Computing methodologies** → **Physical simulation**; **Motion capture**; *Interactive simulation*; *Policy iteration*; *Adversarial learning*.

SIGGRAPH '22 Conference Proceedings, August 7–11, 2022, Vancouver, BC, Canada

ACM ISBN 978-1-4503-9337-9/22/08…$15.00
https://doi.org/10.1145/3528233.3530719

KEYWORDS

Character Animation, Compliant Control, Impedance Control, Admittance Control, Stiffness, Deep Reinforcement Learning, Generative Adversarial Imitation Learning

ACM Reference Format:
Seunghwan Lee, Phil Sik Chang, and Jehee Lee. 2022. Deep Compliant Control. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Proceedings (SIGGRAPH '22 Conference Proceedings), August 7–11, 2022, Vancouver, BC, Canada.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3528233.3530719

1 INTRODUCTION

Humans interact with their surroundings every day. Opening doors, pushing boxes, and playing sports are examples of these interactions. Forces serve as a medium for such interactions, as humans interact with the environment by repeating a dynamic process of sensing these forces and actuating their bodies. In this process, compliance is one of the primary principles which determines human movements during interactions. Compliance provides humans the ability to move in various ways to avoid large amounts of contact forces and adapt to unexpected situations.

This paper aims to build a framework for the simulation and control of humanoids, allowing the humanoid character to interact with their surroundings in physically reliable ways. We can generate a broad spectrum of movements ranging from passive reactions to external physical perturbations, to active manipulations with clear intentions. Technical challenges include defining and modeling of compliance, reproducing realistic human movements, and providing robust control of motor skills for under-actuated dynamical systems. We define the character's compliance from a mechanical point of view. Our motion controller deforms the character to increase

compliance. We also present a learning framework based on Deep Reinforcement Learning (DRL).

Recent progress in DRL has shown its robustness in learning motor skills for movements of high-dimensional characters such as humanoids in physics environments [Lee et al. 2021; Park et al. 2019; Peng et al. 2018; Won et al. 2020]. A control policy (a.k.a controller) learned from DRL allows the character to generate physically plausible motions. The core idea of these methods stems from *imitation learning*. The reference motion data provides the character on how to move, and the controller actuates its character bodies based on their physical states to track the reference motion data in the given environment. While the existence of the reference motion data enables the imitation of human tasks, imitation learning uses position control such as PD control to generate control forces in order to robustly reproduce motions, which can produce stiff and unnatural motions especially when external pertubative forces are being applied. We present a new two-level hierarchical control algorithm based on DRL which allows the character to flexibly interact and move with its surroundings. The compliant controller perturbs the reference motion to achieve compliance, while the imitation controller learns to generate control forces to mimic the modified motions under the physics environment. The resulting controller behaves in a compliant and natural way during the perturbations, and still robustly recover its movements when there are no external forces as it regresses to conventional imitation learning. Our two-level controller minimizes the interaction forces and the control torques for the imitations. We demonstrate the effectiveness of the controller with various examples:

- The character can stand, walk and run even when the interaction forces dominantly affect the character. Based on the forces, our controller generates heterogeneous movements where the lower extremities balance the full body while the upper extremities deform in accordance with the forces.
- We can predict how the mass distributions affect full body movements. Upon Newton's third law of motion, two characters behave differently in order to gain each individual compliance.
- Our controller can perform manipulation tasks such as opening doors or balancing a ball on the head. We demonstrate our controller's robustness and capability by adding perturbations to the environment.

2 RELATED WORK

2.1 Force Control

Force control is one of the major topics in robot control and has been thoroughly studied in Robotics over the past four decades. While position-based control such as PD control focuses mainly on robot positioning, force control aims to achieve precise control of the forces applied by end-effectors. The concepts of impedance and admittance are terminologies used to describe the relationship between the manipulator and the environment [Anderson and Spong 1988; Hogan 1985]. A manipulator is considered an admittance when it controls motions to minimize the external forces, whereas a manipulator is considered an impedance when it controls the forces from the motions. Our framework comprises both the impedance control and the admittance control.

One way to minimize the interaction forces is to embed spring-actuated joints so that the joint has inherent elasticity[Pratt and Williamson 1995; Spong 1987]. Another way is to design a controller as a virtual spring-damper system where the virtual forces of the spring and the damper act as elastic actuators [Ott et al. 2010; Whitney 1987]. The control module can be applied to many applications where safety is crucial such as exoskeletal robot control [Anam and Al-Jumaily 2012; Carignan et al. 2009; Li et al. 2016; Nef et al. 2007] and control in contact-rich spaces [Leboutet et al. 2019; Magrini et al. 2015; Martín-Martín et al. 2019]. Recent advances in deep learning have enriched the domain of force control. Learning of vision and haptic feedback generates robot trajectories for contact-rich tasks in unstructured environments [Lee et al. 2019b]. DRL learns to act compliantly with variable PD gains for a gripper and a hopping robot [Bogdanovic et al. 2020]. Despite great successes in force control, studies have mainly focused on fully actuated robots with few rigid links. In this case, there exist closed-form solutions obtainable by solving the differential kinematics, and it is straightforward to compute trajectories for compliance. It is not straightforward to apply control methods to under-actuated characters. We present a two-level architecture to accommodate the bipedal character.

2.2 Physics-based Character Animation and DRL

In Computer Animation, physics-based character animation has gained great attention for reproducing physically reliable movements. Recent progress in DRL has shown its effectiveness for simulation and control of physically simulated character such as humanoids [Clegg et al. 2018; Heess et al. 2017; Ho and Ermon 2016; Liu and Hodgins 2018; Ma et al. 2021; Won et al. 2021; Yu et al. 2018], flying creatures [Won et al. 2018], quadrupeds [Luo et al. 2020] and even underwater animals like octopuses [Min et al. 2019]. Especially for humanoids, imitation-based learning (a.k.a exampled-guided learning), when provided reference motions, has shown state-of-the-art performance [Bergamin et al. 2019; Lee et al. 2019a; Park et al. 2019; Peng et al. 2018, 2021; Won et al. 2020]. Imitation-based learning adopts concepts of position-based control such as PD control to track the motions and as a result, lacks reactions for physical perturbations and interaction forces.

Meanwhile, there has been a series of studies aiming for human reactions. Data-driven approaches with semi-physics allow the character to react to external forces such as hitting or pushing [Arikan et al. 2005; Coros et al. 2010; Zordan and Hodgins 2002]. Push-and-recovery tests are conducted to analyze the stability of the locomotion controller [Lee et al. 2015; Stephens 2007]. Curriculum learning with DRL can be used to generate task-specific human responses [Lee et al. 2021]. Our framework addresses human reactions in terms of compliance and demonstrates various examples that reproduce physically reliable movements.

3 COMPLIANT CONTROL

3.1 Environment

Our framework assumes the character as articulated rigid bodies, which consists of links and connecting joints. We assume the character is an open-chain (a.k.a tree-structured) articulation with the floating root. The dynamic states of the character can be expressed

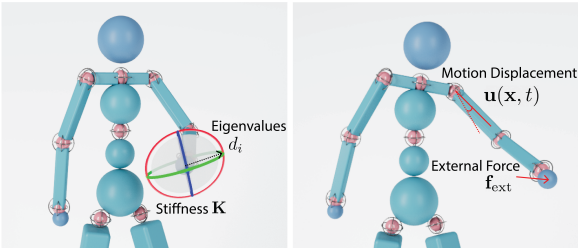

Figure 2: Compliance of the left hand. Due to the geometric locking, the character acts stiffly when applied force is directing to the arm while the other directions have moderate compliance. We detail the interpretation of stiffness matrix in Appendix.

by its joint positions $\mathbf{q} \in \mathbb{R}^n$ and joint velocities $\dot{\mathbf{q}} \in \mathbb{R}^n$ in generalized coordinates. The equations of motion describe the dynamical system as follows:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{c}(\mathbf{q}, \dot{\mathbf{q}}) = \tau + \tau_{\text{ext}} \tag{1}$$

where $\mathbf{M}(\mathbf{q})$ is the mass matrix and $\mathbf{c}(\mathbf{q}, \dot{\mathbf{q}})$ is the Coriolis and gravitational forces. $\tau_{\text{ext}} = \sum_i \mathbf{J}_i^{\mathrm{T}} \mathbf{f}_i$ is the sum of external forces $\mathbf{f}_i$ including contact forces and other external perturbations, where $\mathbf{J}_i$ are Jacobian matrices characterized by the acting point of the force. τ is the control force computed from PD servos for each joint. Given target joint positions $\bar{\mathbf{q}}$, PD servos use proportional and derivative errors to track the target positions:

$$\tau = k_{\text{p}}(\bar{\mathbf{q}} - \mathbf{q}) - k_{\text{v}}\dot{\mathbf{q}} \tag{2}$$

where k_{p} and k_{v} are gains. Given a reference motion $\mathbf{q}^m(t)$, we are able to track the motion by applying the target positions $\bar{\mathbf{q}} = \mathbf{q}^m(t)$. At each time step, the system evolves in time by updating the joint positions and velocities via semi-implicit Euler integration.

3.2 Stiffness

In classical mechanics, *stiffness* is the measurement of how much an object resists deformation in response to an applied force. The mathematical expression of a 1-dof (degrees of freedom) system is defined as $k = -f/x$, where f is the restoring force, and x is the displacement, representing the distance from the resting position due to the applied force. *Compliance* is defined as the inverse of the stiffness $c = k^{-1}$. We can generalize these definitions to high-dimensional characters. Unlike the 1-dof spring, stiffness changes dynamically depending on the pose of the character as well as the acting point of an external force. Given a displacement $\mathbf{x} \in \mathbb{R}^3$ that describes the distance between the current character pose and the reference motion in work space defined by the acting point of the external force, the stiffness of the character is given by:

$$\mathbf{K} = -\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \tag{3}$$

where $\mathbf{K} \in \mathbb{R}^{3\times3}$ is stiffness matrix and $\mathbf{f} \in \mathbb{R}^3$ is a force that the character actuates due to the displacement $\mathbf{x}$. We present the derivation of the matrix $\mathbf{K}$ in Appendix. Note that we derive the character's stiffness in work space rather than joint space, since pertubative forces mostly happens in work space (see Figure 2). We first examine the matrix assuming stationary target positions (i.e. $\bar{\mathbf{q}}$ is constant). As for the PD controlled character, we can explicitly compute Equation (3) by differentiating Equation (2):

$$\mathbf{K} = k_{\text{p}}\mathbf{L} \tag{4}$$

where $\mathbf{L} = (\mathbf{J}\mathbf{J}^{\mathrm{T}})^{-1}$ with the Jacobian matrix $\mathbf{J} = \partial\mathbf{x}/\partial\mathbf{q}$ mapping velocities in joint space to velocities in work space. The matrix $\mathbf{L}$ is the inverse of the Laplacian matrix of the articulation graph of the character weighted by moment arms of each body. Once the external force is applied at a certain point, the mechanical *stiffness* is readily computed through the Jacobian matrix. The stiffness $k = \det(\mathbf{K})$ is computed by taking the determinant of the matrix.

3.3 Modulating Stiffness

ALGORITHM 1: Compliant-induced Motion Controller

Result: $\mathbf{u}(\mathbf{x}, t) = \mathbf{u}_0, \mathbf{u}_1, \cdots$
$\mathbf{u}_0 \leftarrow \mathbf{0}$
for $i = 1, 2, 3, \cdots$ **do**
  $\mathbf{q}_i, \dot{\mathbf{q}}_i \leftarrow$ JointPositions, JointVelocities
  $\dot{\mathbf{u}} \leftarrow \mathbf{0}$
  for $\mathbf{f}_{\text{ext}} \leftarrow \mathbf{f}_0, \mathbf{f}_1, \cdots, \mathbf{f}_j$ **do**
    $\delta\ddot{\mathbf{q}} \leftarrow \text{FD}(\mathbf{q}_i, \dot{\mathbf{q}}_i, \mathbf{f}_{\text{ext}}) - \text{FD}(\mathbf{q}_i, \dot{\mathbf{q}}_i, 0)$: Forward Dynamics
    $\dot{\mathbf{x}} = h\mathbf{J}\delta\ddot{\mathbf{q}}$: Work Space Velocity
    $\dot{\mathbf{u}} = \dot{\mathbf{u}} + \alpha\mathbf{J}^{\mathrm{T}}\mathbf{L}\dot{\mathbf{x}}$: Joint Space Velocity
  end
  $\mathbf{u}_i = \mathbf{u}_{i-1} \oplus h\dot{\mathbf{u}}$: Displacement
end

Since the restoring PD force is a function of $\bar{\mathbf{q}}$, the stiffness matrix also depends on the joint target positions as well as the PD gain k_{p}. We are able to regulate the stiffness to our preference by modulating $\bar{\mathbf{q}}$. We begin by deriving the stiffness matrix given joint target positions:

$$\hat{\mathbf{K}} := k_{\text{p}}[\mathbf{L} - \mathbf{J}^{-\mathrm{T}}\frac{\partial\bar{\mathbf{q}}}{\partial\mathbf{x}}] \tag{5}$$

where $\mathbf{J}^{-1}$ is the pseudo-inverse of the Jacobian matrix. Equation (5) indicates the relation between the stiffness and the joint target positions. If we still keep the joint target positions as $\bar{\mathbf{q}}(t) = \mathbf{q}^m(t)$ independent to the displacement, we can't regulate the stiffness during perturbations to our preference. It is required to design the joint target positions as a function of the displacement. If we generate $\bar{\mathbf{q}}(t)$ such that it meets the condition $\partial\bar{\mathbf{q}}/\partial\mathbf{x} = \mathbf{J}^{\mathrm{T}}\mathbf{L}$, the right hand side of Equation (5) sums to zero, implying that the character is infinitely flexible. Since this means we lose control of the character entirely, we instead take a middle ground. We modify the condition as follows:

$$\frac{\partial\bar{\mathbf{q}}}{\partial\mathbf{x}} = \alpha\mathbf{J}^{\mathrm{T}}\mathbf{L} \tag{6}$$

where $0 \le \alpha \le 1$ is the level of compliance. Incorporating Equation (6) into the Equation (5) results in:

$$\hat{\mathbf{K}} = k_{\text{p}}[\mathbf{L} - \mathbf{J}^{-\mathrm{T}}(\alpha\mathbf{J}^{\mathrm{T}}\mathbf{L})] = k_{\text{p}}(1-\alpha)\mathbf{L} \tag{7}$$

which decreases the stiffness $\hat{k} = \det(\hat{\mathbf{K}}) = (1-\alpha)^3\det(\mathbf{K})$ by the factor of $(1-\alpha)^3$. In our experiments, we choose the level of compliance $\alpha = [0.04, 0.16]$ depending on the magnitude of the forces.

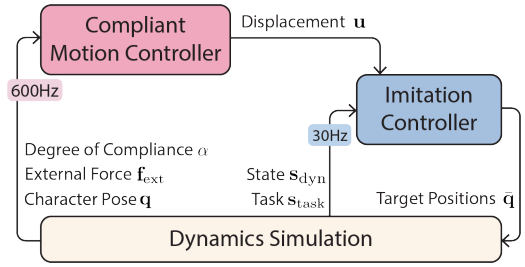

Figure 3: System overview.

It is important to note that we keep the PD gain k_{p} constant even though we are able to regulate the stiffness by decreasing the gain. This is because the decreased gain results in a poor imitation performance. In the following section, we propose a novel motion controller that adjusts motions according to the external forces while robustly tracking the modified motions.

3.4 Compliance-induced Motion Controller

Our goal is to design a controller that generate motions at every time step to react to external perturbations. The controller takes as input the external forces, the character pose, and the level of compliance. The motion controller produces a motion displacement in joint space $\mathbf{u}(\mathbf{x}, t) \in \mathbb{R}^n$ that can be added to the original reference motion by $\bar{\mathbf{q}}(\mathbf{x}, t) = \mathbf{q}^m(t) \oplus \mathbf{u}(\mathbf{x}, t)$ (see Figure 2). The symbol $\oplus$ denotes quaternion multiplications for each joint [Lee 2008]. The dynamic equation for $\mathbf{u}(\mathbf{x}, t)$ such that the generated motions meet the Equation (6) can be written as:

$$\begin{aligned} \dot{\mathbf{u}}(\mathbf{x}, t) &= \alpha \mathbf{J}^{\mathrm{T}} \mathbf{L} \dot{\mathbf{x}}(t) \\ \mathbf{u}(\mathbf{x}, t) &= \int \dot{\mathbf{u}}(\mathbf{x}, t) \mathrm{d}t. \end{aligned} \tag{8}$$

To compute $\dot{\mathbf{x}}(t)$, we first perform forward dynamics of the character by which we obtain the changes of joint accelerations $\delta\ddot{\mathbf{q}}$. Sequentially integrating with time step h and transforming to the work space results in $\dot{\mathbf{x}} = h\mathbf{J}\delta\ddot{\mathbf{q}}$. We update $\mathbf{u}(\mathbf{x}, t)$ by semi-implicit Euler integration. Algorithm 1 shows the overall sequence of the motion controller. If multiple external forces occur, we compute a Jacobian matrix and corresponding joint space velocity for each individual force before summing up to compute $\mathbf{u}(\mathbf{x}, t)$.

For a fully-actuated dynamical system, we can directly control the character by producing control forces with the target joint positions $\bar{\mathbf{q}}(\mathbf{x}, t) = \mathbf{q}^m(t) \oplus \mathbf{u}(\mathbf{x}, t)$. For a humanoid, naïve use of PD control with $\bar{\mathbf{q}}(\mathbf{x}, t)$ loses control of its balance, since the character is under-actuated at the root joint, and there are also additional contact and friction forces that are not considered. To remedy this, a more sophisticated feedback controller is required to modulate the joint target positions in accordance with the character state. In the following section, we present a framework for learning the motor skills for the physically simulated character equipped with our compliance-induced motion controller.

4 LEARNING MOTOR SKILLS

Given the motions generated by the compliant motion controller, we follow the trails of imitation learning approaches to animate the character in physics-based simulation (see Figure 3). We learn an imitation controller (a.k.a control policy) $\pi_\theta(\bar{\mathbf{q}}|\mathbf{s})$ that produces the actual target joint positions so that the character imitates the reference motions $\mathbf{q}^m(t) \oplus \mathbf{u}(\mathbf{x}, t)$ as close as possible according to the character state $\mathbf{s}$. The controller $\pi_\theta(\bar{\mathbf{q}}|\mathbf{s})$ is defined by a deep neural network whose parameters are given by θ. The state $\mathbf{s} = (\mathbf{s}_{\text{dyn}}, \mathbf{u}, \mathbf{s}_{\text{task}})$ includes the dynamic state of the character, the motion displacement of the motion controller, and optionally the state of the tasks.

We use DRL with imitation rewards to learn the controller. In DRL, the controller observes the state $\mathbf{s}_t$ to produce the target joint positions $\bar{\mathbf{q}}_t$ at each time step. $\bar{\mathbf{q}}_t$ is then applied to the system through the PD control, evolving to the next time step $t + 1$. This results in the new state $\mathbf{s}_{t+1}$ and the scalar reward $r_t = R(\mathbf{s}_t, \mathbf{s}_{t+1})$. The reward function encourages the character to mimic the motion, and the controller is optimized its parameters to maximize the expected accumulated reward:

$$\theta^* = \arg\max_\theta \mathbb{E}_{\mathbf{s}_0, \bar{\mathbf{q}}_0, \mathbf{s}_1, \cdots} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] \tag{9}$$

where γ is the discount factor, $\mathbf{s}_0 \sim \rho_0$ is sampled from an initial state distribution ρ_0, $\bar{\mathbf{q}}_t \sim \pi_\theta(\mathbf{s}_t)$ is sampled from the controller, and $\mathbf{s}_{t+1} \sim p(\mathbf{s}_t, \bar{\mathbf{q}}_t)$ is sampled from transition probability p, which specifies the dynamics of the environment. Since evaluation of the objectives requires an intractable computational cost, DRL also learns another network called the value function $V^\pi(\mathbf{s})$ [Lillicrap et al. 2015; Mnih et al. 2013]. During each epoch of learning, the controller $\pi_\theta(\bar{\mathbf{q}}|\mathbf{s})$ and the value function $V^\pi(\mathbf{s})$ are jointly optimized using transition tuples $(\mathbf{s}_t, \bar{\mathbf{q}}_t, r_t, \mathbf{s}_{t+1})$. We use Proximal Policy Optimization, which uses the surrogate loss of the objective to improve the training robustness by clipping the probability ratio of the controller output.

4.1 Imitation Reward with Adversarial Network

The beauty of DRL is the ability to learn complex motor skills with the scalar reward function. The reward function r encourages the character to imitate the motion, and optionally guides the character to achieve various task objectives. The reward is designed as:

$$r = w_{\text{imit}} r_{\text{imit}} + w_{\text{task}} r_{\text{task}} \tag{10}$$

where r_{imit} is an imitation reward which evaluates how closely the character tracks the reference motion, r_{task} is a task-specific reward, and $w_{\text{imit}} = 0.5$ and $w_{\text{task}} = 0.5$ are their weights. The compliant motion controller describes not only the joint angles but also the movements of the root joint. The imitation reward r_{imit} is defined by the multiplication of two terms:

$$r_{\text{imit}} = r_q r_p \tag{11}$$

where r_q is the pose reward and r_p is the positional reward. The pose reward measures a discrepancy between the simulated character pose and the reference pose, and the positional reward encourages the character to move in accordance with the root displacement $\mathbf{u}_p$ extracted from $\mathbf{u}$. The root displacement denotes the deviation of the character from the original motion in terms of the root position. To track the root displacement encouraged by r_p, it is required to learn diverse motions including standing, walking, and

running, along with transitions between these motions. If an appropriate motion planner such as a motion graph [Won et al. 2020], an RNN-based motion generator [Park et al. 2019], a motion matching [Bergamin et al. 2019], or a deep-learning-based high-level controller [Peng et al. 2017] provides motions that match the root displacements, our compliant controller can be applied directly without any modifications in the rewards. We further explain this case in Appendix. However, constructing such motion planners require annotating and organizing the reference motions along with parameter tuning, which can be tedious. Alternatively we utilize a generative adversarial network to learn motion planning from unorganized data set as well as motor control for the imitation at once. We train an adversarial motion network to discriminate the simulated character pose from the reference motion distribution. Once trained, the network can evaluate how similar the character pose is to the reference data, acting as the imitation reward function. The ability to learn the motion distribution enables the controller to actively plan goal-driven maneuvers along the distribution of the reference data without using any pre-generated trajectories.

To learn the adversarial motion network, we follow the trails of generative adversarial imitation learning, which stems from behavior cloning [Ho and Ermon 2016; Peng et al. 2021; Torabi et al. 2018]. We first extract expert trajectories from the motion dataset $\mathcal{M}$. The trajectories are represented as pairs of the current and next state $(\mathbf{s}, \mathbf{s}') \sim \mathcal{M}$ in a per-frame basis. While learning the controller, the controller π generates controller trajectories $(\mathbf{s}, \mathbf{s}') \sim \pi$. The adversarial network $D_\psi(\mathbf{s}, \mathbf{s}') \in \mathbb{R}$ is optimized to discriminate the controller trajectories from the expert trajectories:

$$\min_\psi \left[\mathbb{E}_{(\mathbf{s},\mathbf{s}')\sim\mathcal{M}}(D_\psi(\mathbf{s},\mathbf{s}')-1)^2 + \mathbb{E}_{(\mathbf{s},\mathbf{s}')\sim\pi}(D_\psi(\mathbf{s},\mathbf{s}')+1)^2 \right] \quad (12)$$

where we use a least-square loss to prevent the gradient vanishing problem [Peng et al. 2021]. The pose reward function r_q is given by:

$$r_q(\mathbf{s},\mathbf{s}') = 1 - \beta(D(\mathbf{s},\mathbf{s}')-1)^2 \quad (13)$$

where β is the shape factor. The collaboration of Equation (12) and (13) encourages the controller to produce the current and next state pairs $(\mathbf{s}, \mathbf{s}')$ that are indistinguishable from the expert trajectory's distribution. When the expert dataset is small, the discriminator learns to discern expert trajectories from the controller's motion too quickly, such that the discriminator reward becomes unhelpful for the controller to imitate the dataset. In this case we use $\beta = 0.2$ to give an offset reward whereas we use $\beta = 0.25$ for large datasets. The positional reward r_p is designed to match the positional displacement:

$$r_p = w_p \exp\left(-\sigma_p \|\mathbf{p} - \mathbf{u}_p\|^2\right) + w_v \exp\left(-\sigma_v \|\mathbf{v} - \dot{\mathbf{u}}_p\|^2\right) \quad (14)$$

where $\mathbf{p}$ and $\mathbf{v}$ are the position and the linear velocity of the root body respectively. $\mathbf{u}_p$ and $\dot{\mathbf{u}}_p$ are the positional displacement of the root joint and its velocity respectively, extracted from $\mathbf{u}$. In our experiments, the weights are $w_p = 0.7$, $w_v = 0.3$, $\sigma_p = 0.5$, and $\sigma = 1.0$.

4.2 State

Similar to previous work, we specify the state [Lee et al. 2019a; Ma et al. 2021; Park et al. 2019; Peng et al. 2018; Won et al. 2020]. The dynamic state $\mathbf{s}_{\text{dyn}}$ is expressed in terms of positions, orientations, linear velocities, and angular velocities of each body. The orientations are encoded using rotation matrices. The positions and the velocities are expressed in the character's local coordinate. The local coordinate is defined as the root body position projected onto the ground and the root body orientation aligned with the xz-plane. Overall, the state dimension is 306. We represent the state in a velocity-free manner to easily incorporate the motion displacement into the state features. We include positions of all bodies at the current character pose and the previous character pose. The positions are expressed in the character's current local coordinate.

As the imitation controller learns to displace its bodies according to the motion displacement $\mathbf{u}$, the controller trajectories deviate from the expert motion trajectories by the amount of motion displacement $\mathbf{u}$. Learning the displaced expert motions $\mathbf{q}^m \oplus \mathbf{u}$ for all $m \in \mathcal{M}$ requires numerous sampling of the current and next state pairs $(\mathbf{s}, \mathbf{s}') \sim \mathbf{q}^m \oplus \mathbf{u}$ and evaluating again the adversarial motion network for all $\mathbf{u}$. It is intractable to shift the expert motion distribution directly. We rather shift the state of the simulated character by the opposite direction of $\mathbf{u}$. We encode the state $\mathbf{s} = \mathbf{s}(\mathbf{q} \ominus \mathbf{u})$ where $\ominus$ computes the joint-wise quaternion differences $\mathbf{q} \ominus \mathbf{u} := (\log(\mathbf{u}_0^{-1}\mathbf{q}_0), \log(\mathbf{u}_1^{-1}\mathbf{q}_1), ..., \log(\mathbf{u}_n^{-1}\mathbf{q}_n))$, where n is the number of joints. By comparing the expert trajectories and the shifted state, we can construct an imitation reward that accomodates compliance.

5 RESULTS

5.1 Environment and Controller Settings

Our simulation is written in C++. We use the open source library DART to simulate articulated rigid bodies [Lee et al. 2018]. Our character consists of 12 ball-and-socket joints and 2 welded joints. The character is 163cm tall and weighs 65kg. The shape of each body is modeled by a sphere or a box to compute the inertia tensor and to detect collisions between the character and environments. We use PD gains $k_\text{p} = 300$ to $500\ N \cdot m/rad$ and $k_\text{v} = 2\sqrt{k_\text{p}}$ for critical damping. Stable PD control is used to increase simulation stability [Tan et al. 2011]. Our simulation frequency is 600Hz, but when the external forces are large enough that it affects the simulation's stability, we simulate the dynamics at a higher rate of 900Hz.

The compliant motion controller starts with zero displacement and is integrated at the same rate as the physics simulation. We insert the external forces we are interested in reacting compliantly into the input of the compliant controller. Once the interaction forces are detected, we prepare the Jacobian matrix to compute the stiffness matrix. Since each column of the Jacobian refers to how much each joint can move, we sometimes give weights to the columns to generate task-dependent plausible motions. Specifically we give more weights for the hip joints and the shoulder joints because the range of motions of those joints are wider than the other joints. We omit root rotational and y-axis translational displacements since these terms often produce physically invalid positional goals. When there are no interaction forces, the spring-damper system is activated on the motion displacement to recover the original motions.

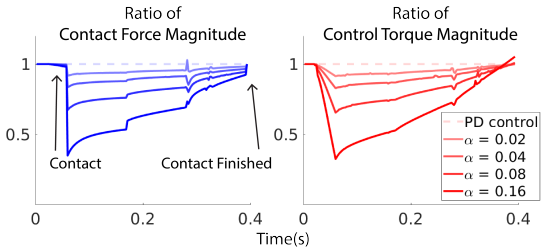

Figure 4: Levels of compliance. All values are normalized by the values of PD controlled character without any compliance. (Left) The ratio of contact force magnitude when the character collides with an obstacle. (Right) The ratio of control torque magnitude.

Our imitation controller is based on a deep neural network. We use the open source framework PyTorch for learning the neural network [Paszke et al. 2017]. We stack three fully connected layers for the controller $\pi_\theta(\mathbf{a}|\mathbf{s})$ as well as the value function $V^\pi(\mathbf{s})$. Each layer has 256 nodes and is initialized using Xavier initialization, and no dropout is used in our experiments. We use Proximal Policy Optimization(PPO) to learn the imitation controller. We use $\lambda = 0.95$ for the Generalized Advantage Estimate(GAE) and the discount factor $\gamma = 0.95$. During learning, a Gaussian noise for each action space is used for exploration, and we set the standard deviations to be 6°. Whenever 2048 state transition tuples $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ are collected, we update the parameters of the controller using a stochastic gradient descent method at learning rate 10^{-5} with a minibatch size of 128. We also use Reference State Initialization proposed by Peng et al. [2018] to acquire well-distributed samples in terms of the reference motions. The learning takes 6 to 24 hours with 20 to 80 million tuples for challenging tasks such as performing a backflip. We use a Ryzen 3950x CPU equipped with MPI-based parallelization to simulate the physics environment and collect the state transition tuples. NVIDIA 2070Ti GPU is used to accelerate the training of neural networks.

We also train a discriminator to evaluate the reward function in DRL. The discriminator and the imitation controller are updated jointly. The discriminator consists of three fully connected layers with 256 nodes. To learn the discriminator, we sample the controller trajectories $(\mathbf{s}, \mathbf{s}') \sim \pi$ from the simulator as well as the expert trajectories from the reference motions $(\mathbf{s}, \mathbf{s}') \sim \mathcal{M}$. We sample an equal amount of tuples $(\mathbf{s}, \mathbf{s}')$ from the simulation and the dataset in order for the discriminator to learn two distributions in a balanced manner. We use a minibatch size of 16 with learning rate 10^{-4}. In addition to the loss function (12), we add a gradient penalty loss term and a regularization loss term for the parameters of the last layer to enhance learning stability [Gulrajani et al. 2017].

5.2 Level of Compliance

We choose the level of compliance $\alpha = [0.04, 0.16]$ depending on tasks and interaction forces (see Table in Appendix). To determine a good starting point for this value, we conduct a simple experiment for our motion control with various levels of compliance. In this experiment, the character is pushed by obstacles and the character learns to maintain its balance. The obstacles weigh 1kg and are thrown in various directions with initial velocities $[2.0, 3.0]m/s$. During contacts, both the magnitude of contact forces and magnitude of torques required for PD control decrease as the level of compliance increases (see Figure 4). We observe that the character's joint angles produce an implausible range of motion when $\alpha > 0.2$. In our later experiments, we mostly use $\alpha = 0.16$. When the interaction forces become larger, we decrease the level of compliance until it does not exceed the range of motion.

5.3 Learning Motor Skills

We learn assorted motor skills with our compliant controller. We use reference motions available on the web such as CMU motion dataset and *Mixamo*. The motion clips are retargeted to our character using *Autodesk MotionBuilder*™. Some examples are learned from multiple reference motions, and the controller can learn transition among them. We further use a rich set of motion data set that helps response and recovery movements.

Hand-in-hand running is a task where two characters run together hand in hand. The hands are attached using ball-and-socket joints and thus constraint forces are continuously applied along the hands. We take the constraint forces as input for the motion controller. We prepare a pulling character which moves kinematically. We do not simulate the pulling character, but it serves as a kinematic constraint for the simulated character. While adhering to the kinematic constraint, the simulated character successfully learns to take additional steps to prevent losing its balance. In addition, the compliant control reduces the interaction forces. For a side-by-side comparison, we show a character that learns motor skills without using our compliant motion control. Since small changes in movement create drastically changing constraint forces, the character moves dominantly by the constraint forces and produces stiff movements.

Chicken Fight is a sport where two or more players hop on one leg and bump the opponent players until they fall over. We use 17.6 seconds of hopping motion which moves in all directions. We learn two players with different mass distributions. One weighs 85kg and the other weighs 36kg. We modulate PD gains according to the different masses. We put all the contact forces between two characters into our motion controller. During training of each character, we replace the opponent as a box-shaped punching bag that has the same mass. According to Newton's third law of motion, they receive the same magnitude of forces when bumping. Since the mass distribution affects the forward dynamics and stiffness, the light character is pushed out farther away than the heavy character to compensate for the changes in momentum.

Trampoline is a device consisting of stretchable fabric attached to a steel frame. Humans leverage spring forces of the fabric to bounce on the trampoline. We learn motor skills on the trampoline ranging from balancing to backflip. To simplify the simulation, the trampoline is modeled as a rigid plate with linear actuators at each corner. We vertically bounce the character every 3 seconds. We activate our compliant controller after the character hits the ground for a second to make absorb-and-recovery movements. Interestingly, our compliant controller dissipates the character's total energy, meaning that the character absorbs the impact during contact and reduces fluctuation of the trampoline (see Figure 5).

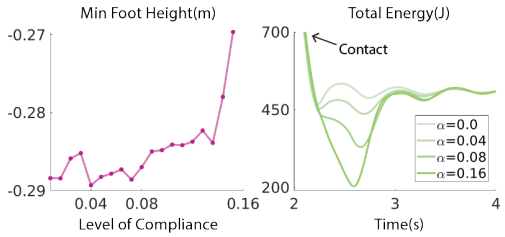

Figure 5: The *Trampoline* example. (Left) Minimum foot height during contact. Higher value indicates lower fluctuation of the trampoline. (Right) Total energy.

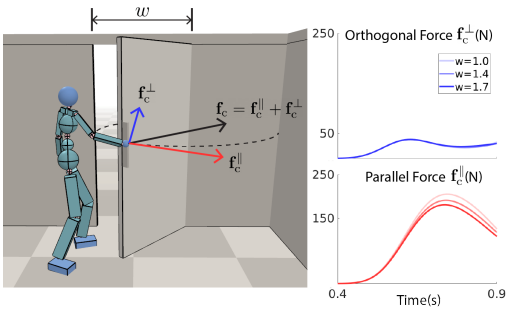

Figure 6: *Opening doors* example. (Left) The constraint force can be decomposed to orthogonal and parallel components relative to the knob, shown in blue and red arrows. (Up Right) The orthogonal force with the varying door size. (Bottom Right) The parallel force.

5.4 Manipulation

The character also learns to manipulate objects. During manipulation, it is crucial to repeat a dynamic process of sensing interaction forces and actuating bodies to adjust to the situations. Since manipulation tasks require much more active control from the character, we drive the character by designing additional task-specific reward functions.

Opening doors are one of the common physical interactions in daily life. Humans open a door by grabbing and pushing a knob. Since the door consists of a hinge, the manipulability of the knob forms a circle whose center is on the hinge. To open the door, the end-effector of the character should be on the circle. While forces acting parallel to the knob's normal direction works to open the door, forces acting perpendicular to the normal direction hinder the character from acting compliantly (see Figure 6). In our experiment, the character learns to open doors with varying door width w. We prepare a 5-second motion clip of opening a door, and we add a phase variable $\phi \in [0, 1]$ to the state to learn the phase-aware controller, which acts as the controller of Peng et al. [2018]. When the hand is close to the knob, the hand is temporarily glued to the door using a zero-dof joint constraint. We take the orthogonal term of the constraint force as an input for the compliant control. The state for the task encodes the angle of the door and the relative position of knob expressed in the end-effector coordinate $\mathbf{s}_{\text{task}} = (\phi, \mathbf{p}_{\text{knob}})$. The task reward is designed as $r_{\text{task}} = \exp(-1.5 \max(0, \hat{\phi} - \phi))$ where $\hat{\phi} = 0.7$ is target door angle. After training, the character is able to deal with arbitrary size of the door ranging $w \in [1.0, 1.7]$ even though it only learns for the fixed size door $w = 1.0$ during learning. The compliant control successfully maintains the orthogonal forces consistently low independent of the door size while the character modulates the parallel forces according to the door size.

Balancing a ball is another example where the character learns to hold a ball on a rod on the character's head. Unlike other examples where the character passively receives forces, balancing the ball requires active procedures that produce forces to manipulate the ball as it wants. To do so, we add additional 6-DOF variables in the control space to add fictional forces at the head and root joint. These fictional forces are not used at all in simulation, but is used to calculate the displacements in the compliant controller, which allows the controller to actively deform the body to its preference. We encode the state for the task $\mathbf{s}_{\text{task}} = (\mathbf{p}_{\text{ball}}, \mathbf{v}_{\text{ball}}, \mathbf{p}_{\text{rod}}, \mathbf{v}_{\text{rod}})$ where $\mathbf{p}_{\text{ball}}, \mathbf{v}_{\text{ball}}$ are the position and the linear velocity of the ball and $\mathbf{p}_{\text{rod}}, \mathbf{v}_{\text{rod}}$ are the position and the linear velocity of the rod. These are expressed in the character's local coordinate. The task reward $r_{\text{task}} = \exp(-2.0||\text{proj}_{xz}(\mathbf{p}_{\text{ball}} - \mathbf{p}_{\text{head}})||^2)$ is designed to minimize the discrepancy between the position of the ball and the head projected onto the ground. As a result, the character actively deforms the pose and steps to move its position, mimicking the balance control of an inverted pendulum on a cart.

6 DISCUSSION

Deep reinforcement learning with imitation rewards has gained great attention in physics-based character animation. Our framework enables reinforcement learning to address both long-term plans for imitation and short-term plans for compliance. The compliant controller reduces both the interaction forces and the control torques necessary for the balance and tasks. Moreover, our framework has the ability to learn manipulation tasks, which require active alternations of given reference motion sequences.

Our framework bridges the gap between position control and force control. Position control such as PD control is thought to be an indispensable module for the control of a high-dimensional floating-base character. Major drawbacks of using position control arise when the character interacts with its surroundings. The character applies a large amount of force to the environment, which potentially causes simulation instabilities or undesirable movements. Meanwhile, studies related to force control in Robotics have aimed on fixed-base low dimensional manipulators such as gripper robots. Our framework generalizes the concept of force control in terms of the character's compliance to incorporate force control into DRL. Combining extrinsic imitation of the human movements with intrinsic measurement of compliance allows the character to seamlessly learn both position control and force control.

Our framework also has limitations. We modulated the character's stiffness based only on external physical interactions with the environment. As a result, when there are no interactions, our method becomes a baseline imitation learning framework that doesn't provide any compliance to the imitated motions. To achieve compliance in the general case, one way is to modulate PD variables including gains as well as target joint positions according to

the motions and the character's physical states. However, a noticeable trade-off exists between performing imitation (or high-level goals) and achieving compliance, as we have observed that the controller gradually fails to imitate the reference motion as we minimize the torque actuation. Another limitation is the way of defining compliance. We assume that it only relies on the character pose and its articulations, which are geometric properties of the character. Human compliance is more complex than just geometry and articulation. For example, humans employ three strategies for maintaining their balance: ankle strategy, ankle-hip strategy, and stepping strategy. These strategies use extra information about the dynamics of the system, such as the center of pressure and base of support, which are not incorporated into our compliant control. Lastly, the method requires manual tuning of the level of compliance and the weights of the Jacobian matrix columns. It is feasible to tune the parameters by hand since the dimension of the parameters is relatively low. A straightforward expansion of our framework would consider anisotropic and joint-wise properties, which dramatically increases the dimension of hyperparameters. It might be possible to find these parameters from the motion dataset to reproduce realistic human movements.

ACKNOWLEDGMENTS

This work was supported by *Samsung Research Funding Center* under Project Number SRFC-IT1801-01.

REFERENCES

Khairul Anam and Adel Ali Al-Jumaily. 2012. Active exoskeleton control systems: State of the art. *Procedia Engineering* 41 (2012), 988–994.

Robert J Anderson and Mark W Spong. 1988. Hybrid impedance control of robotic manipulators. *IEEE Journal on Robotics and Automation* 4, 5 (1988), 549–556. https://doi.org/10.1109/56.20440

Okan Arikan, David A Forsyth, and James F O'Brien. 2005. Pushing people around. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 59–66.

Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. 2019. DReCon: Data-Driven Responsive Control of Physics-Based Characters. *ACM Transactions on Graphics* 38, 6, Article 206 (2019).

Miroslav Bogdanovic, Majid Khadiv, and Ludovic Righetti. 2020. Learning variable impedance control for contact sensitive tasks. *IEEE Robotics and Automation Letters* 5, 4 (2020), 6129–6136.

Craig Carignan, Jonathan Tang, and Stephen Roderick. 2009. Development of an exoskeleton haptic interface for virtual task training. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 3697–3702.

Alexander Clegg, Wenhao Yu, Jie Tan, C. Karen Liu, and Greg Turk. 2018. Learning to Dress: Synthesizing Human Dressing Motion via Deep Reinforcement Learning. *ACM Transactions on Graphics* 37, 6, Article 179 (2018).

Stelian Coros, Philippe Beaudoin, and Michiel Van de Panne. 2010. Generalized biped walking control. *ACM Transactions On Graphics (TOG)* 29, 4 (2010), 1–9.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. 2017. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028* (2017).

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

Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. *Advances in neural information processing systems* 29 (2016), 4565–4573.

Neville Hogan. 1985. Impedance Control: An Approach to Manipulation: Part I—Theory. *Journal of Dynamic Systems, Measurement, and Control* 107, 1 (1985), 1–7.

Quentin Leboutet, Emmanuel Dean-Leon, Florian Bergner, and Gordon Cheng. 2019. Tactile-Based Whole-Body Compliance With Force Propagation for Mobile Manipulators. *IEEE Transactions on Robotics* 35, 2 (2019), 330–342. https://doi.org/10.1109/TRO.2018.2889261

Jehee Lee. 2008. Representing Rotations and Orientations in Geometric Computing. *IEEE Computer Graphics and Applications* 28, 2 (2008), 75–83.

Jeongseok Lee, Michael X. Grey, Sehoon Ha, Tobias Kunz, Sumit Jain, Yuting Ye, Siddhartha S. Srinivasa, Mike Stilman, and C. Karen Liu. 2018. DART: Dynamic Animation and Robotics Toolkit. *The Journal of Open Source Software* 3, 22 (2018), 500.

Michelle A. Lee, Yuke Zhu, Krishnan Srinivasan, Parth Shah, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Jeannette Bohg. 2019b. Making Sense of Vision and Touch: Self-Supervised Learning of Multimodal Representations for Contact-Rich Tasks. In *2019 International Conference on Robotics and Automation (ICRA)*. 8943–8950. https://doi.org/10.1109/ICRA.2019.8793485

Seyoung Lee, Sunmin Lee, Yongwoo Lee, and Jehee Lee. 2021. Learning a family of motor skills from a single motion clip. *ACM Transactions on Graphics* 40, 4 (2021), 1–13.

Seunghwan Lee, Moonseok Park, Kyoungmin Lee, and Jehee Lee. 2019a. Scalable Muscle-Actuated Human Simulation and Control. *ACM Transactions on Graphics* 38, 4, Article 73 (2019).

Yoonsang Lee, Kyungho Lee, Soon-Sun Kwon, Jiwon Jeong, Carol O'Sullivan, Moon Seok Park, and Jehee Lee. 2015. Push-recovery stability of biped locomotion. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 1–9.

Zhijun Li, Zhicong Huang, Wei He, and Chun-Yi Su. 2016. Adaptive impedance control for an upper limb robotic exoskeleton using biological signals. *IEEE Transactions on Industrial Electronics* 64, 2 (2016), 1664–1674.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

Libin Liu and Jessica Hodgins. 2018. Learning Basketball Dribbling Skills Using Trajectory Optimization and Deep Reinforcement Learning. *ACM Trans. Graph.* 37, 4, Article 142 (July 2018), 14 pages. https://doi.org/10.1145/3197517.3201315

Ying-Sheng Luo, Jonathan Hans Soeseno, Trista Pei-Chun Chen, and Wei-Chao Chen. 2020. Carl: Controllable agent with reinforcement learning for quadruped locomotion. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 38–1.

Li-Ke Ma, Zeshi Yang, Tong Xin, Baining Guo, and KangKang Yin. 2021. Learning and Exploring Motor Skills with Spacetime Bounds. *Computer Graphics Forum* 40, 2 (2021).

Emanuele Magrini, Fabrizio Flacco, and Alessandro De Luca. 2015. Control of generalized contact motion and force in physical human-robot interaction. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2298–2304. https://doi.org/10.1109/ICRA.2015.7139504

Roberto Martín-Martín, Michelle A. Lee, Rachel Gardner, Silvio Savarese, Jeannette Bohg, and Animesh Garg. 2019. Variable Impedance Control in End-Effector Space: An Action Space for Reinforcement Learning in Contact-Rich Tasks. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 1010–1017. https://doi.org/10.1109/IROS40897.2019.8968201

Sehee Min, Jungdam Won, Seunghwan Lee, Jungnam Park, and Jehee Lee. 2019. SoftCon: Simulation and Control of Soft-Bodied Animals with Biomimetic Actuators. *ACM Trans. Graph.* 38, 6, Article 208 (Nov. 2019), 12 pages. https://doi.org/10.1145/3355089.3356497

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

Tobias Nef, Matjaz Mihelj, Gabriela Kiefer, Christina Perndl, Roland Muller, and Robert Riener. 2007. ARMin-Exoskeleton for arm therapy in stroke patients. In *2007 IEEE 10th international conference on rehabilitation robotics*. IEEE, 68–74.

Christian Ott, Ranjan Mukherjee, and Yoshihiko Nakamura. 2010. Unified Impedance and Admittance Control. In *2010 IEEE International Conference on Robotics and Automation*. 554–561. https://doi.org/10.1109/ROBOT.2010.5509861

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* 38, 6, Article 205 (2019).

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.

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* 37, 4, Article 143 (2018).

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* 36, 4, Article 41 (2017).

Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. 2021. AMP: Adversarial Motion Priors for Stylized Physics-Based Character Control. *ACM Transactions on Graphics* 40, 4, Article 1 (July 2021).

G.A. Pratt and M.M. Williamson. 1995. Series elastic actuators. In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, Vol. 1. 399–406 vol.1. https://doi.org/10.1109/IROS.1995.525827

M. W. Spong. 1987. Modeling and Control of Elastic Joint Robots. *Journal of Dynamic Systems, Measurement, and Control* 109, 4 (12 1987), 310–318. https://doi.org/10.1115/1.3143860 arXiv:https://asmedigitalcollection.asme.org/dynamicsystems/article-pdf/109/4/310/5604812/310_1.pdf

Benjamin Stephens. 2007. Humanoid push recovery. In *2007 7th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 589–595.

Jie Tan, Karen Liu, and Greg Turk. 2011. Stable proportional-derivative controllers. *IEEE Computer Graphics and Applications* 31, 4 (2011), 34–44.

Faraz Torabi, Garrett Warnell, and Peter Stone. 2018. Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158* (2018).

Daniel E Whitney. 1987. Historical perspective and state of the art in robot force control. *The International Journal of Robotics Research* 6, 1 (1987), 3–14.

Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2020. A Scalable Approach to Control Diverse Behaviors for Physically Simulated Characters. *ACM Transactions on Graphics* 39, 4, Article 33 (2020).

Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2021. Control strategies for physically simulated characters performing two-player competitive sports. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–11.

Jungdam Won, Jungnam Park, and Jehee Lee. 2018. Aerobatics Control of Flying Creatures via Self-Regulated Learning. *ACM Trans. Graph.* 37, 6, Article 181 (Dec. 2018), 10 pages. https://doi.org/10.1145/3272127.3275023

Wenhao Yu, Greg Turk, and C. Karen Liu. 2018. Learning Symmetric and Low-Energy Locomotion. *ACM Trans. Graph.* 37, 4, Article 144 (July 2018), 12 pages. https://doi.org/10.1145/3197517.3201397

Victor Brian Zordan and Jessica K Hodgins. 2002. Motion capture-driven simulations that hit and react. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 89–96.