ADHERENT: Learning Human-like Trajectory Generators for Whole-body Control of Humanoid Robots

Paolo Maria Viceconte[1,2,*] **Raffaello Camoriano**[3] **Giulio Romualdi**[1,4] **Diego Ferigo**[1,5]
Stefano Dafarra[1] **Silvio Traversaro**[1] **Giuseppe Oriolo**[2] **Lorenzo Rosasco**[3,4,6] **Daniele Pucci**[1,5]

[1]Artificial and Mechanical Intelligence, Istituto Italiano di Tecnologia, Genoa, Italy
[2]DIAG, Sapienza Università di Roma, Roma, Italy
[3]Laboratory for Computational and Statistical Learning - IIT@MIT, Istituto Italiano di Tecnologia, Genoa, Italy, and Massachusetts Institute of Technology, Cambridge, MA, USA
[4]DIBRIS, Università degli Studi di Genova, Genova, Italy
[5]Machine Learning and Optimisation, University of Manchester, Manchester, UK
[6]Center for Brains, Minds and Machines, MIT, Cambridge, MA, USA
[*]paolo.viceconte@iit.it - viceconte@diag.uniroma1.it

Abstract

Human-like trajectory generation and footstep planning has been a longstanding open problem in humanoid robotics. Meanwhile, research in computer graphics kept developing machine-learning methods for character animation based on training human-like models directly on motion capture data. Such methods proved effective in virtual environments, mainly focusing on trajectory visualization. This paper presents ADHERENT, a system architecture integrating machine-learning methods used in computer graphics with whole-body control methods employed in robotics to generate and stabilize *human-like* trajectories for humanoid robots. Leveraging human motion capture locomotion data, ADHERENT yields a general footstep planner, including forward, sideways, and backward walking trajectories that blend smoothly from one to another. At the joint configuration level, ADHERENT computes data-driven whole-body postural references coherent with the generated footsteps, thus increasing the *human likeness* of the resulting robot motion. Extensive validations of the proposed architecture are presented with both simulations and real experiments on the iCub humanoid robot. Supplementary video: https://sites.google.com/view/adherent-trajectory-learning.

1 Introduction

The general problem of generating trajectories for humanoid robots still remains a challenge for the robotics community. The complexity of the problem increases considerably when targeting real-time trajectory generation for different environmental conditions and robot locomotion modes. For instance, whole-body trajectory generation methods for robot walking soon become numerically intractable due to the high dimensionality of the problem, especially when the overall generated motion is required to fulfill a certain degree of *human likeness*. We propose a system architecture for efficiently addressing whole-body *human-like* trajectory generation for humanoid robots. The architecture leverages recent research in computer graphics (CG) targeting character animation via learning-based methods [1–5] (see Appendix, Sec. A.2). We focus on Mixture of Experts (MoE) methods such as Phase-Functioned Neural Networks (PFNN) [1] and Mode-Adaptive Neural Networks (MANN) [2]. We integrate the latter with state-of-the-art hierarchical whole-body humanoid robot control methods, which proved effective on a diverse set of real-world humanoids [6–13] (see Appendix, Sec. A.1). Learning-based methods notably improve generality and human likeness of trajectory generation, while whole-body hierarchical controllers provide the reliability and robustness required in the real world.

NeurIPS 2021 Workshop on Robot Learning: Self-Supervised and Lifelong Learning, Virtual, Virtual

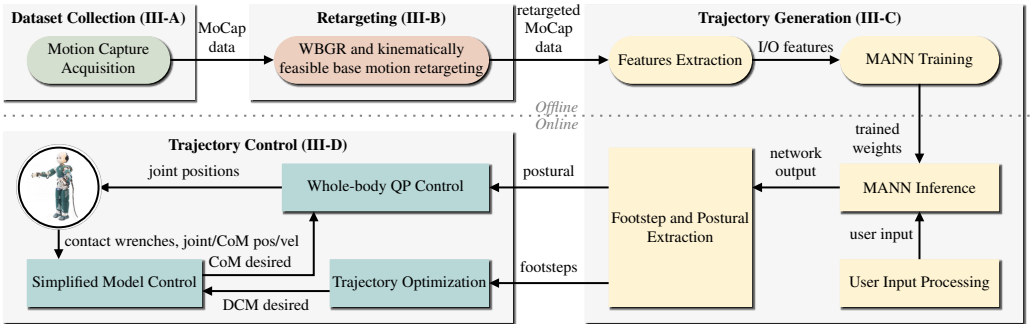

Figure 1: Block diagram of the overall learning-based ADHERENT architecture proposed in this work.

Main contributions: **i)** We present ADHERENT (humAn-Driven wHolE-body REference geNerator and conTroller), the first step towards a comprehensive learning-based architecture for efficient *human-like* whole-body trajectory generation and control of humanoid robots, and validate its robustness with extensive simulations and real-world experiments on the iCub humanoid; **ii)** We demonstrate the generality of the learning-based footstep planner incorporated into ADHERENT by showing its adaptability to diverse walking patterns, facing directions, start and stops, and smooth transitions among these; **iii)** We verify the improved human likeness of the whole-body motions exploiting the data-driven postural references provided by ADHERENT along with the footstep plan.

2 ADHERENT

The proposed ADHERENT architecture consists of four main components: *Dataset Collection*, *Retargeting*, *Trajectory Generation* and *Trajectory Control* – see Fig. 1. In the following, we present the methods implementing each component. In light of ADHERENT's modularity, specific methods can be easily replaced by more efficient and effective ones in future instances of the architecture.

2.1 Dataset Collection

The *Dataset Collection* component is in charge of acquiring human locomotion data. For this, we use our human wearable data processing framework [14, 15] that fuses data from a sensorized suit by XSens technologies [16], carrying 17 wireless inertial sensors scattering the entire body. Data span a wide range of walking motions (forward, backward, lateral, diagonal) performed on a flat terrain with continuously-changing steering direction. Stops and restarts are included in the sequences, characterized by steps of variable length. Please refer to the Appendix (Sec. C.1) for further details.

2.2 Retargeting

The *Retargeting* component adjusts the human trajectories so as when the modified trajectories are applied to the robot, its motion turns out to be *similar* to the human one. We retarget the collected motion capture (MoCap) data by enhancing the Whole-Body Geometric Retargeting (WBGR) technique (see Appendix, Sec. B.2) with a kinematically-feasible base motion retargeting that renders the robot base motion compatible with the retargeted joint trajectories (See Appendix, Sec. C.2).

2.3 Trajectory Generation

We interactively generate trajectories for the robot by exploiting the MANN architecture detailed in the Appendix (Sec. B.3). The training dataset preparation and network structure are inspired by [2].

Features extraction The input and output vectors for MANN are extracted by preprocessing the retargeted MoCap dataset. The input $\mathbf{x}_i$ at time step t_i includes the robot state at t_{i-1} and the ground base trajectory data at t_i, i.e., past and future base trajectory data projected on the ground, subsampled to obtain 12 data points equally spaced on a 2 s window centered at t_i. The output $\mathbf{y}_i$ at time step t_i includes the robot state at t_i, the ground angular base transformation from t_{i-1} to t_i, and the future ground base trajectory data at t_{i+1} (consisting of 6 data points equally spaced on a 1 s window starting from t_i). Further details on the input and output vectors are included in the Appendix (Sec. C.3).

Network structure The MANN architecture used in this work is composed of a Motion Prediction Network and a Gating Network with 3 hidden layers of 512 and 32 units each, respectively. ELU activation [17] is used. The Gating Network receives the full input $\mathbf{x}_i$. We use $K = 4$ experts.

User input processing At inference time, the user provides via joypad two continuous signals to interactively generate trajectories: i) the *motion direction*: the direction in which the user wants the robot to move; and ii) the *facing direction*: the direction towards which the user wants the robot to align the mean of its base and torso horizontal pointing directions. At fixed facing direction, varying the motion direction allows to switch between frontal, sideways, and backward walking. At fixed motion direction, varying the facing direction allows steering. Moreover, releasing the analog for the motion direction leads the robot to a stop. A step-by-step description of the user inputs processing, which is critical for the predictive performances of MANN, is provided in the Appendix (Sec. C.4).

Network output postprocessing The network output $\mathbf{y}_i$ is used to update the robot configuration. In particular, the joint position $\boldsymbol{s}_i$ included in $\mathbf{y}_i$ becomes the new joint configuration and the ground angular base velocity $\dot{b}_i^a$ is exploited to update the base orientation. The base position is instead updated by applying the same kinematic feasibility procedure used at the retargeting stage (Sec. 2.2).

Footstep and postural extractor The desired feet positions and orientations composing the footstep plan are retrieved from the generated trajectory by the *Footstep Extractor*. In particular, a new foot position is added to the plan once the support foot changes. Concerning orientations, in the case of flat terrain the plan only requires the predicted yaw angle of the support foot, while roll and pitch angles are set to zero. The joint positions $\boldsymbol{s}_i$ included in the network prediction $\mathbf{y}_i$ constitute a whole-body human-like postural. However, having been trained on a MoCap dataset collected at 60 fps, the network generates references only compatible with such frequency. Still, the *whole-body Quadratic Programming (QP) control* layer may require posturals at a different frequency. The *Postural Extractor* interpolates the network's predictions to obtain posturals at the required frequency.

2.4 Trajectory Control

We execute on the robot the walking trajectories by leveraging the three-layer control architecture described in the Appendix (Sec. B.4). Given the footstep plan provided by the *Footstep Extractor*, a reference Divergent Component of Motion (DCM) [18] trajectory is obtained by the planner included in the *Trajectory Optimization* layer. In particular, we adopt the implementation from [19], providing a feasible DCM trajectory even for variable Center of Mass (CoM) height. Given a desired DCM trajectory, at each control cycle the *Simplified Model Control* layer computes the desired CoM velocity $\dot{\boldsymbol{x}}^*$ by concatenating Eq.(1) and Eq.(2) from the Appendix (Sec. B.4). Then, the desired CoM position $\boldsymbol{x}^*$ is retrieved by Euler integration. A QP problem of the form defined in Eq.(3) from the Appendix (Sec. B.4) is formulated from the postural returned by the *Postural Extractor*. The desired feet poses and CoM trajectory $\{\dot{\boldsymbol{x}}^*, \boldsymbol{x}^*\}$ are set as hard constraints. An additional soft constraint aims to zero the chest roll and pitch angles. Finally, the desired joint velocity $\dot{\boldsymbol{s}}^*$ included in the solution of the QP problem is integrated, and the resulting desired joint position $\boldsymbol{s}^*$ is sent to the robot.

3 Results

Here, we report the results obtained after training our architecture on the processed MoCap dataset. MANN is trained in a classical regression setting, minimizing the mean squared error between the ground truth and the network prediction. The Appendix also includes training details (Sec. D.1) and additional analyses on the robustness (Sec. D.2) and blending coefficients activation (Sec. D.3).

Trajectory generation The *Trajectory Generation* component interactively generates walking trajectories. Each prediction requires around 3 ms on a 9-th generation Intel Core i7 CPU @ 2.60 GHz. By varying motion and facing directions, the user can move the robot forward, backward, and sideways in a human-like fashion. Changes in the input signals promptly translate into smooth transitions between different walking patterns. By releasing the motion direction stick, the user can stop the robot and then restart the motion at will. Fig. 2 (top) shows a complex trajectory, including several walking patterns and smooth transitions between them. The footstep positions extracted from the entire trajectory are visualized in red and blue for the right and left foot, respectively. A larger variety of trajectory generations is reported in the supplementary video.

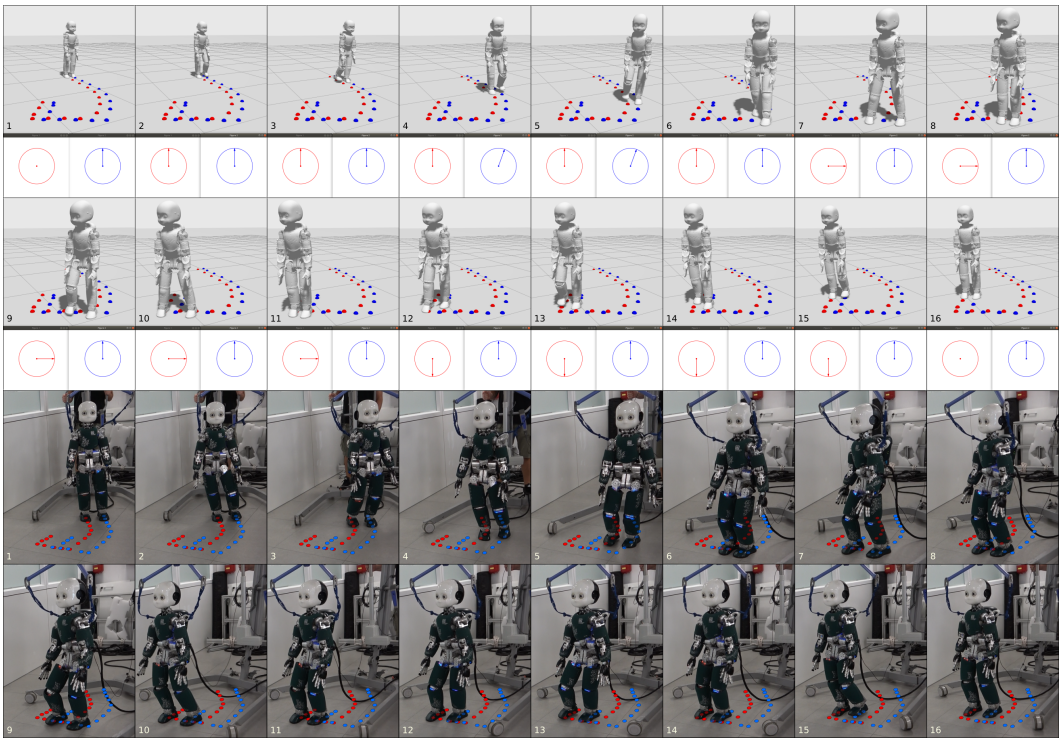

Figure 2: *Top*: A mixed trajectory including forward (1-3), right-oriented forward (4-6), right-side (7-11), and backward (12-15) walking, as well as smooth transitions between them, and a final stop (16). Below each frame, the user inputs interactively shaping the trajectory are plotted from the local viewpoint of the simulated robot (red: Desired motion direction; blue: Desired facing direction). *Bottom*: The very same trajectory on iCub.

Trajectory control The generated trajectories are executed on the real-world 32-Degree of Freedom (DoF) iCub humanoid robot [20], which is 104 cm tall and weighs approximately 33 Kg. The control architecture composed by the *Simplified Model Control* and *Whole Body QP Control* layers runs at 100 Hz on a 4-th generation Intel Core i7 @ 1.7 GHz. Fig. 2 (bottom) illustrates the successful execution of the complex trajectory whose generation is shown in the upper part of the same figure. The footstep sequence performed by the robot is added to the visualization for the sake of clarity. The execution of this trajectory, along with others, is also presented in the supplementary video.

Human likeness We evaluate the *human likeness* of the trajectories executed by ADHERENT on the real-world iCub robot. We compare them with walking trajectories adopting a fixed postural for the upper body, as is often the case in classical humanoid robot locomotion. A side-by-side comparison in the case of a forward walking is shown in the supplementary video. As it can be observed, the overall motion with ADHERENT postural shows an improved *human likeness*. Additional considerations on the degree of human-likeness achieved with ADHERENT are included in the Appendix (Sec. D.4).

4 Discussion

ADHERENT is able to generate reference trajectories in real time (3 ms per prediction step), thanks to efficient neural-network-based feedforward prediction. Therefore, it achieves comparable efficiency with respect to simplified-model trajectory generators, while being able to produce more general high-dimensional whole-body trajectories that state-of-the-art methods can only generate offline due to excessive computational complexity. We also demonstrate that ADHERENT-generated trajectories can be successfully executed on an advanced humanoid robot whose physical properties significantly differ from those of the human body. Such successfully-executed trajectories robustly cover a broad range of motion types and steering capabilities. Improving *human likeness* of the robot motion is an additional feature enabled by ADHERENT. We show that learning-driven *human-like* motions generated by the network can be successfully transferred to the real robot.

References

[1] Daniel Holden, Taku Komura, and Jun Saito. Phase-functioned neural networks for character control. *ACM Transactions on Graphics*, 2017.

[2] He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. Mode-adaptive Neural Networks for Quadruped Motion Control. *ACM Transactions on Graphics*, 2018.

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

[4] Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. DReCon: data-driven responsive control of physics-based characters. *ACM Transactions on Graphics*, 2019.

[5] Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. Learning Agile Robotic Locomotion Skills by Imitating Animals. In *Robotics: Science and Systems*, 2020.

[6] Siyuan Feng, Eric Whitman, X Xinjilefu, and Christopher G. Atkeson. Optimization based full body control for the atlas robot. In *Humanoids*, 2014.

[7] Hongkai Dai, Andrés Valenzuela, and Russ Tedrake. Whole-body motion planning with centroidal dynamics and full kinematics. In *Humanoids*, 2014.

[8] Alexander Herzog, Nicholas Rotella, Stefan Schaal, and Ludovic Righetti. Trajectory generation for multi-contact momentum control. In *Humanoids*, 2015.

[9] Justin Carpentier, Steve Tonneau, Maximilien Naveau, Olivier Stasse, and Nicolas Mansard. A versatile and efficient pattern generator for generalized legged locomotion. In *ICRA*, 2016.

[10] Stefano Dafarra, Gabriele Nava, Marie Charbonneau, Nuno Guedelha, Francisco Andrade, Silvio Traversaro, Luca Fiorio, Francesco Romano, Francesco Nori, Giorgio Metta, and Daniele Pucci. A Control Architecture with Online Predictive Planning for Position and Torque Controlled Walking of Humanoid Robots. In *IROS*, 2018.

[11] Giulio Romualdi, Stefano Dafarra, Yue Hu, and Daniele Pucci. A Benchmarking of DCM Based Architectures for Position and Velocity Controlled Walking of Humanoid Robots. In *Humanoids*, 2018.

[12] George Mesesan, Johannes Englsberger, Gianluca Garofalo, Christian Ott, and Alin Albu-Schäffer. Dynamic Walking on Compliant and Uneven Terrain using DCM and Passivity-based Whole-body Control. In *Humanoids*, 2019.

[13] N Ramuzat, G Buondonno, S Boria, and Olivier Stasse. Comparison of Position and Torque Whole Body Control Schemes on the TALOS Humanoid Robot. 2021.

[14] Claudia Latella, Yeshasvi Tirupachuri, Lorenzo Rapetti, Diego Ferigo, Silvio Traversaro, Ines Sorrentino, Francisco Javier Andrade Chavez, Francesco Nori, and Daniele Pucci. A Human Wearable Framework for Physical Human-Robot Interaction. In *I-RIM*, 2019.

[15] Lorenzo Rapetti, Yeshasvi Tirupachuri, Kourosh Darvish, Claudia Latella, and Daniele Pucci. Model-Based Real-Time Motion Tracking using Dynamical Inverse Kinematics on SO(3). *Algorithms*, 2020.

[16] Daniel Roetenberg, Henk Luinge, and Per Slycke. Xsens MVN: Full 6DOF Human Motion Tracking Using Miniature Inertial Sensors. 2013.

[17] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *ICLR*, 2016.

[18] Johannes Englsberger, Christian Ott, and Alin Albu-Schaffer. Three-dimensional bipedal walking control using Divergent Component of Motion. In *IROS*, 2013.

[19] Giulio Romualdi, Stefano Dafarra, Giuseppe L'Erario, and Daniele Pucci. Non-Linear DCM Trajectory Optimization with Variable Center of Mass Height. *I-RIM*, 2020.

[20] Giorgio Metta, Lorenzo Natale, Francesco Nori, Giulio Sandini, David Vernon, Luciano Fadiga, Claes von Hofsten, Kerstin Rosander, Manuel Lopes, José Santos-Victor, Alexandre Bernardino, and Luis Montesano. The iCub humanoid robot: An open-systems platform for research in cognitive development. *Neural Networks*, 2010.

Appendix

A Related Work Details

A.1 Humanoid Robot Locomotion

State-of-the-art architectures for humanoid locomotion simplify the whole-body trajectory generation problem by hierarchically decomposing it into several layers [1]. Layers' functionalities can be categorized [2] in: i) *Trajectory optimization*, providing a high-level footstep plan given user input; ii) *Simplified model control*, computing feasible CoM trajectories given the footsteps; and iii) *Whole-body QP control*, producing dynamically-feasible joint trajectories. Instead of directly optimizing over large configuration spaces, the first two layers tend to use simplified models to compute solutions. For instance, the *unicycle planner* [3] employs a unicycle model to produce footstep plans at the trajectory optimization layer, constraining the plan to simple directed walking on a plane [4], [5].

In recent years, this class of hierarchical architectures has been successfully applied to produce robust walking on a diverse range of complex humanoids [1, 6–8, 3, 2, 9, 10], also allowing for the integration of reactive strategies [11, 12]. However, simplified models do not fully represent the complex humanoid mechanical structure in order to reduce computational cost and allow for on-line operation. As a result, they restrict the attainable solutions set and the resulting behaviors with respect to those achieved by humans. In particular, they cannot efficiently compute walking patterns with unconstrained footstep placement. Moreover, whole-body human likeness is hard to explicitly encode and optimize for with respect to other attributes such as feasibility, stability, and robustness, and is therefore usually neglected in such schemes.

Data-driven models of human trajectories have recently been explored to enable human-like behavior in robotics [13, 14]. Applications include anticipatory trajectory generation for human-robot collaboration [15]. Still, such methods are focused on overall path planning (i.e., CoM trajectory), and do not target human likeness at the joint or footstep level.

A.2 Character Animation in Computer Graphics

It is important to note that the problem of human-like trajectory generation is not limited to robotics research. Indeed, it is a prominent topic in CG research too, especially due to applications to realistic character animation, and has witnessed several recent breakthroughs based on the introduction of machine-learning methods. The core of the problem can be framed as the kinematic prediction of the whole-body joints configuration in the next time step, given the current configuration and the high-level target trajectory to be followed (i.e., obtained from human input).

Many works approach this problem by modeling it as a nonlinear autoregressive model with exogenous inputs. They employ powerful learning-based predictive models able to capture the motion's complexity in high dimensions. In PFNN [16], the predictive model is a phase-weighted mixture of neural networks trained on human motion capture data. At prediction time, the network weights are blended according to a cyclic phase function encoding the periodicity of the walking motion. This resulted in a significant breakthrough for character control, enabling remarkably natural motion and smooth transitions. However, training data need to be annotated with phase function values, which can be costly or unfeasible for complex and non-periodic motions. In MANN [17], the latter problem is solved by substituting the fixed phase function with a *gating network*, which learns end-to-end how to effectively blend the network weights. Note that both PFNN and MANN are limited to trajectory generation for kinematic rendering only. In fact, their target applications are in settings in which natural visual appearance, rather than dynamic control of a real-world system, is the primary requirement (i.e., videogames).

In [18], *Motion Matching* is employed alongside reinforcement learning (RL) to retrieve motions from a MoCap dataset and provide them as references to train a policy in simulation. Improvements in terms of memory usage were proposed in [19]. DeepMimic [20] employs MoCap data to guide policy training via an imitation reward component. However, although having demonstrated remarkable capabilities in simulation and on real quadrupeds [21], RL approaches are severely limited by substantial inefficiencies and are yet to be successfully applied to real-world humanoids.

B Background

B.1 Notation

Please refer to the following notation for the quantities introduced in the remaining of the Appendix:

- $\mathcal{I}$ and $\mathcal{B}$ denote the inertial frame and the base frame of the robot. In the specific case of iCub [22], $\mathcal{B}$ is positioned at the level of the waist, in between the two legs, with the X axis pointing backward and the Z axis upwards.
- Given two frames $\mathcal{A}$ and $\mathcal{C}$, ${}^{\mathcal{A}}R_{\mathcal{C}} \in SO(3)$ represents the rotation matrix between the frames, i.e., given two vectors ${}^{\mathcal{A}}\boldsymbol{p}, {}^{\mathcal{C}}\boldsymbol{p} \in \mathbb{R}^3$ respectively expressed in $\mathcal{A}$ and $\mathcal{C}$, the rotation matrix ${}^{\mathcal{A}}R_{\mathcal{C}}$ is such that ${}^{\mathcal{A}}\boldsymbol{p} = {}^{\mathcal{A}}R_{\mathcal{C}}\,{}^{\mathcal{C}}\boldsymbol{p}$.
- Superscripts $\cdot^{\mathcal{H}}$ and $\cdot^{\mathcal{R}}$ indicate quantities referring to the human and the robot, respectively.
- The $m \times m$ identity and zero matrices are denoted by I_m and 0_m, respectively.
- When referring to network inputs $\mathbf{x}$, outputs $\mathbf{y}$, weights $\hat{\boldsymbol{\alpha}}$, and blending coefficients $\boldsymbol{\theta}$ or to their elements, subscript $\cdot_i$ indicates quantities of the i-th time step t_i.
- The vec$(\cdot)$ operator vectorizes matrices by rows.
- Given $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^3$, we define $\boldsymbol{a}^{\wedge} = A \in \mathbb{R}^{3\times 3}$ as the skew-symmetric matrix such that $\boldsymbol{a}^{\wedge}\boldsymbol{b} = \boldsymbol{a} \times \boldsymbol{b}$.
- n denotes the robot's DoFs.
- $\boldsymbol{\nu} = ({}^{\mathcal{I}}\dot{\boldsymbol{p}}_{\mathcal{B}}, {}^{\mathcal{I}}\boldsymbol{\omega}_{\mathcal{B}}, \dot{\boldsymbol{s}}) \in \mathbb{R}^{6+n}$ is the generalized velocity of the complete floating-base system, where ${}^{\mathcal{I}}\boldsymbol{\omega}_{\mathcal{B}}$ is the angular velocity of the base frame w.r.t. the inertial frame, whose coordinates are expressed in the inertial frame, i.e., ${}^{\mathcal{I}}\dot{R}_{\mathcal{B}} = {}^{\mathcal{I}}\boldsymbol{\omega}_{\mathcal{B}}^{\wedge}\,{}^{\mathcal{I}}R_{\mathcal{B}}$.

B.2 Whole-Body Geometric Retargeting

Among the various approaches to human motion retargeting (see, e.g., [23–26]), WBGR is a recent method that is easily adaptable to different robot models and human subjects [27]. Assuming a degree of topological similarity between the human's and the robot's mechanical structures, WBGR makes use of m correspondences between the frames associated with m human and robot links at a reference configuration. Then, given the human link orientations ${}^{\mathcal{I}}R^i_{\mathcal{H}}$, $i \in 1, ..., m$ to be retargeted onto the robot, WBGR allows to retrieve the robot joint angles by solving the inverse kinematics problem with the robot orientations ${}^{\mathcal{I}}R^i_{\mathcal{R}} = {}^{\mathcal{I}}R^i_{\mathcal{H}}\,{}^{\mathcal{H}}R^i_{\mathcal{R}}$ as targets: each ${}^{\mathcal{H}}R^i_{\mathcal{R}}$ is a proper constant rotation matrix accounting for possible human-robot frame misalignment.

B.3 Mode-Adaptive Neural Networks

MANN is a recently-proposed neural network architecture for responsive character motion generation specifically designed for multi-modal and unlabeled data [17]. In particular, assume that $\mathbf{x}_i$ encodes the previous configuration of the controlled character as well as the desired future motion specified by the user. Then, MANN predicts a new configuration $\mathbf{y}_i$ for the character that achieves the user-specified motion. The next user input is combined with $\mathbf{y}_i$, forming the next autoregressive network input $\mathbf{x}_{i+1}$. This enables MANN to iteratively generate trajectories following the MoCap data distribution while being responsive to the user. The main characteristic of this architecture, which builds upon the *Mixture of Experts* paradigm [28], is that of being composed of two subnetworks:

- The *Motion Prediction Network*: given $\mathbf{x}_i$, it predicts $\mathbf{y}_i$;
- The *Gating Network*: given $\mathbf{x}_i$ or a subsampled input $\hat{\mathbf{x}}_i$, it predicts the blending coefficients vector $\boldsymbol{\theta}_i = [\theta_{i1}, ..., \theta_{iK}]^{\top}$ used to dynamically compute the weights vector $\hat{\boldsymbol{\alpha}}_i$ of the Motion Prediction Network from the K expert network weights vectors $\{\boldsymbol{\alpha}_1, ..., \boldsymbol{\alpha}_K\}$.

In an end-to-end training procedure from unstructured MoCap data, both the weights $\boldsymbol{\mu}$ of the Gating Network and the K expert weights $\{\boldsymbol{\alpha}_1, ..., \boldsymbol{\alpha}_K\}$ are learned. At runtime, the weights $\hat{\boldsymbol{\alpha}}_i$ of the Motion Prediction Network at time step i are dynamically computed by linearly combining the K experts $\{\boldsymbol{\alpha}_1, ..., \boldsymbol{\alpha}_K\}$ with the blending coefficients $\boldsymbol{\theta}_i$ predicted by the Gating Network, that is, $\hat{\boldsymbol{\alpha}}_i = \sum_{j=1}^{K} \theta_{ij}\boldsymbol{\alpha}_j$.

Table 1: Breakdown of the MoCap data.

Walking motion	Duration [min]	Frames	Stops
Forward	8.2	29.500	25
Backward	9	32.450	25
Side	9.45	34.000	27
Diagonal	4.2	15.125	16
Mixed	30.48	109.710	75

B.4 A Three-layer Control Architecture for Humanoid Robot Locomotion

A state-of-the-art control architecture for humanoid robot locomotion relevant for this work is composed of three nested layers that exploit both simplified and complete robot models [2]. Given the footsteps, in the outer *trajectory optimization* loop, an exponential interpolation technique is used to plan a desired DCM trajectory, smoothed via a third-order polynomial during the double support phases. Then, the central *simplified model control* loop is in charge of stabilizing the DCM dynamics by using the Zero Moment Point (ZMP) position $\boldsymbol{r}^{zmp} \in \mathbb{R}^2$ as control input. The tracking of the desired DCM position and velocity $\boldsymbol{\xi}_{ref}, \dot{\boldsymbol{\xi}}_{ref} \in \mathbb{R}^2$ is guaranteed by the instantaneous control law given by:

$$\boldsymbol{r}_{ref}^{zmp} = \boldsymbol{\xi}_{ref} - \frac{\dot{\boldsymbol{\xi}}_{ref}}{\bar{\omega}} + K_p^{\xi}(\boldsymbol{\xi} - \boldsymbol{\xi}_{ref}) + K_i^{\xi} \int (\boldsymbol{\xi} - \boldsymbol{\xi}_{ref}) dt, \tag{1}$$

where $K_p^{\xi} > I_2$, $K_i^{\xi} > 0_2$, $\bar{\omega} = \sqrt{g/z_0}$, g is the gravitational constant and z_0 denotes the constant CoM height assumed for the Linear Inverted Pendulum (LIP) model [29]. The desired ZMP position $\boldsymbol{r}_{ref}^{zmp}$ is then stabilized along with the reference ground CoM position and velocity $\boldsymbol{x}_{ref}, \dot{\boldsymbol{x}}_{ref} \in \mathbb{R}^2$ by means of the control law given by:

$$\dot{\boldsymbol{x}}^* = \dot{\boldsymbol{x}}_{ref} - K_{zmp}(\boldsymbol{r}_{ref}^{zmp} - \boldsymbol{r}^{zmp}) + K_{com}(\boldsymbol{x}_{ref} - \boldsymbol{x}), \tag{2}$$

where $K_{com} > \bar{\omega} I_2$ and $0_2 < K_{zmp} < \bar{\omega} I_2$. Finally, the inner *whole-body QP control* loop computes the robot velocity $\boldsymbol{\nu}$ as the solution to a stack of tasks formulation with hard and soft constraints, cast as a QP problem of the form:

$$\underset{\boldsymbol{\nu}}{min} \;\; \frac{1}{2}\boldsymbol{\nu}^T H \boldsymbol{\nu} + \boldsymbol{g}^T \boldsymbol{\nu} \;\; s.t. \;\; A_c \boldsymbol{\nu} = \boldsymbol{b}_c, \;\; \dot{\boldsymbol{s}}^- \leq \dot{\boldsymbol{s}} \leq \dot{\boldsymbol{s}}^+, \tag{3}$$

where H and $\boldsymbol{g}$ are evaluated from a low-priority postural task (soft constraint), while A_c and $\boldsymbol{b}_c$ are retrieved from the chosen high-priority tasks (hard constraints) and the final inequalities encode constraints on the maximum joint velocity. The joint velocity from the solution of the above problem, obtained via standard QP solvers, is directly integrated to get joint positions for position control.

C ADHERENT

C.1 Dataset Collection Details

As detailed in Table 1, each kind of motion in the collected dataset is performed for several minutes in a row. Then, plenty of transitions between different motions are collected in a long mixed sequence. Our final dataset comprises around 1 h of unlabeled MoCap data at 60 fps. We then double it by mirroring, i.e., for each data point the base orientation is mirrored with respect to the world X-Z plane, while the left and right link orientations for the limbs are switched and mirrored with respect to the human model's mid-sagittal plane, resulting in a total of 441570 data points.

C.2 Kinematically-feasible Base Motion Retargeting

WBGR in [27] does not address the base motion retargeting, i.e., the retargeted base position and orientation may not be compatible with the robot kinematics, thus possibly leading to a robot moving forward faster than what its walking pace entails. In other words, a swaying effect arises when dynamic motions are retargeted to robot models structurally different from the human subject [27].

While in CG generating models which fit the collected data is a viable workaround [17] [18], base motion retargeting for actual robots requires special attention.

First, assume that: i) the robot makes at least one known contact with the environment; and ii) each foot is modeled as a rectangular patch. Then, we propose the following approach for kinematically-feasible base motion retargeting:

1. The contact point ${}^{\mathcal{I}}\boldsymbol{p}_c$ is identified as the lowest among the 8 vertices of the feet's rectangular approximations;
2. The retargeted base orientation ${}^{\mathcal{I}}R_{\mathcal{B}}$ is directly retrieved from the MoCap data;
3. The retargeted base position ${}^{\mathcal{I}}\boldsymbol{p}_b$ is computed by forward kinematics from ${}^{\mathcal{I}}\boldsymbol{p}_c$, constrained to remain fixed between two consecutive retargeting steps, via ${}^{\mathcal{I}}\boldsymbol{p}_b = {}^{\mathcal{I}}\boldsymbol{p}_c + {}^{\mathcal{I}}R_{\mathcal{C}}{}^{\mathcal{C}}\boldsymbol{p}_b$, where $\mathcal{C}$ is the frame attached to the contact point (i.e., the frame placed in the lowest vertex and oriented as the support foot) and ${}^{\mathcal{C}}\boldsymbol{p}_b$ is the base position, expressed in $\mathcal{C}$, computed by forward kinematics in the updated joint configuration returned by the latest WBGR iteration.

As a result, we obtain retargeted motions that resemble human ones at the links level and are kinematically feasible at the base level, as shown in the supplementary video.

C.3 Features Extraction Details

In this work, the input vector $\mathbf{x}_i$ for MANN is defined as follows:

$$\mathbf{x}_i = \{\underbrace{\text{vec}(P_i),\ \text{vec}(D_i),\ \text{vec}(V_i),\ l_i,}_{\text{Base trajectory}}\ \underbrace{\boldsymbol{s}_{i-1},\ \dot{\boldsymbol{s}}_{i-1}}_{\text{Robot state}}\} \in \mathbb{R}^{137}, \tag{4}$$

where $P_i = \{\boldsymbol{p}_i^1, ..., \boldsymbol{p}_i^{12}\} \in \mathbb{R}^{2\times 12}$ are ground base positions, $D_i = \{\boldsymbol{d}_i^1, ..., \boldsymbol{d}_i^{12}\} \in \mathbb{R}^{2\times 12}$ are ground facing directions (i.e., mean of base and chest pointing directions, projected on the ground), $V_i = \{\boldsymbol{v}_i^1, ..., \boldsymbol{v}_i^{12}\} \in \mathbb{R}^{2\times 12}$ are ground base velocities, $l_i = \sum_{j=7}^{12} \left\| \boldsymbol{p}_i^j - \boldsymbol{p}_i^{j-1} \right\| \in \mathbb{R}$ is the length of the future ground trajectory and $\boldsymbol{s}_{i-1}, \dot{\boldsymbol{s}}_{i-1} \in \mathbb{R}^{32}$ are joint positions and velocities at t_{i-1}.

The output vector $\mathbf{y}_i$ is instead defined as follows:

$$\mathbf{y}_i = \{\underbrace{\text{vec}(P_{i+1}), \text{vec}(D_{i+1}), \text{vec}(V_{i+1})}_{\text{Future base trajectory}}, \underbrace{\boldsymbol{s}_i, \dot{\boldsymbol{s}}_i}_{\substack{\text{Robot}\\ \text{state}}}, \underbrace{\dot{b}_i^a}_{\substack{\text{Base}\\ \text{transf.}}}\} \in \mathbb{R}^{101}, \tag{5}$$

where $P_{i+1} = \{\boldsymbol{p}_{i+1}^1, ..., \boldsymbol{p}_{i+1}^6\} \in \mathbb{R}^{2\times 6}$ are future ground base positions, $D_{i+1} = \{\boldsymbol{d}_{i+1}^1, ..., \boldsymbol{d}_{i+1}^6\} \in \mathbb{R}^{2\times 6}$ are future ground facing directions, $V_{i+1} = \{\boldsymbol{v}_{i+1}^1, ..., \boldsymbol{v}_{i+1}^6\} \in \mathbb{R}^{2\times 6}$ are future ground base velocities, $\boldsymbol{s}_i, \dot{\boldsymbol{s}}_i \in \mathbb{R}^{32}$ are joint positions and velocities at t_i, while $\dot{b}_i^a = \beta_i / \Delta t_i \in \mathbb{R}$, with $\Delta t_i = t_i - t_{i-1}$, and β_i denoting the angle between the ground facing directions at t_i and t_{i-1} (i.e., $\boldsymbol{d}_i^6$ and $\boldsymbol{d}_{i-1}^6$, respectively).

All the ground base trajectory data in $\mathbf{x}_i$ and $\mathbf{y}_i$ are expressed in the bidimensional local reference frame defined by the ground base position at t_i (i.e., $\boldsymbol{p}_i^6$) and the ground facing direction at t_i (i.e., $\boldsymbol{d}_i^6$) along with its orthogonal vector. By stacking all the input and output vectors resulting from the processing above, we obtain the input and output matrices $X \in \mathbb{R}^{441570\times 137}$ and $Y \in \mathbb{R}^{441570\times 101}$ which, normalized to have zero mean and unit variance, constitute our training set.

C.4 User Input Processing Details

The desired motion and facing directions specified by the user are visualized in Fig. 3 (left), from the local viewpoint of a robot proceeding forward while steering left. Details on how these inputs are actually transferred to the network follow, since the user inputs represent an external factor, never seen by the network during training but directly shaping its input $\mathbf{x}_i$ at runtime.

The user-specified motion and facing directions are smoothly interpolated to generate a desired future ground base trajectory $\{P_{i+1}^*, D_{i+1}^*, V_{i+1}^*\}$ whose components are defined as in Eq.(5). In particular, the user-specified motion direction is used to define the last point of a quadratic Bézier curve starting from $\boldsymbol{p}_i^6$ and constrained to end on the asymmetric shape shown in black in Fig. 3. We obtain P_{i+1}^*

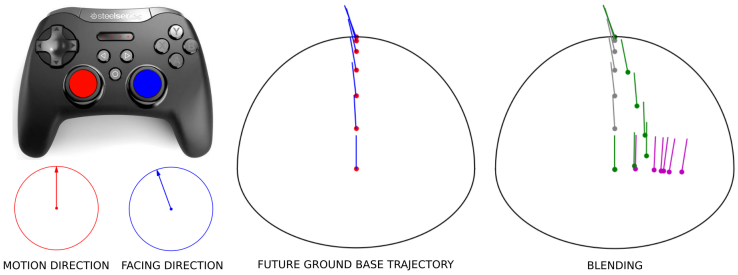

Figure 3: The desired motion and facing directions from the joypad (left) define the desired future ground base trajectory (center). On the right, the user-specified future trajectory (grey) is blended with the future trajectory from the previous network prediction (magenta), leading to the desired future ground base trajectory (green) actually included in the next input to the network.

by subsampling 6 data points from this Bézier curve. As a result of the asymmetric constraint, P^*_{i+1} is longer for forward rather than sideways or backward walking. The user-specified facing direction is instead mapped into a series of facing directions D^*_{i+1} progressively driving the current value to the desired one. V^*_{i+1} is obtained by differentiating P^*_{i+1}. Fig. 3 (center) provides a visualization, from the local robot's viewpoint, of P^*_{i+1} (red dots) and D^*_{i+1} (blue vectors) generated from the user-specified inputs (left).

As a last step, $\{P^*_{i+1}, D^*_{i+1}, V^*_{i+1}\}$ is blended with the $\{P_{i+1}, D_{i+1}, V_{i+1}\}$ retrieved from the previous output $\mathbf{y}_i$ in order to obtain the ground base trajectory data for the next input $\mathbf{x}_{i+1}$. For the blending, an example of which is shown in Fig. 3 (right), we follow the method proposed in [16]. For instance, the desired future ground base positions P^*_{i+1} are blended with the future ground positions outputed by the network P_{i+1} via:

$$Blend(P_{i+1}, P^*_{i+1}, t, \tau_p) = (1 - t^{\tau_p})\, P_{i+1} + t^{\tau_p}\, P^*_{i+1} \tag{6}$$

where t goes from 0 to 1 as ground base positions go towards the future (i.e. towards the limit of the considered 1-second future window) and $\tau_p = 1.5$ controls the responsiveness of the character to the user-specified inputs. The very same procedure is employed to blend the future facing directions and base velocities, with τ_p in Eq. (6) respectively replaced by $\tau_d = 1.3$ and $\tau_v = 1.3$.

Note that user-specified inputs may result in a desired motion which is absent or rare in the training dataset. In such case (e.g., when the user requests too abrupt steering) the network may generate unexpected motions. We solve this issue by limiting the local variation of facing direction that the user can require to $45°$ and $20°$ for forward and backward/sideways walking, respectively.

C.5 Network Output Postprocessing Details

When the user tries to stop the robot, a small in-place rotation persists. Indeed, given an $\mathbf{x}_i$ corresponding to a desired stop, the network predicts a $\mathbf{y}_i$ whose $\dot{b}^a_i$ component is slightly different from zero. We solve this issue by imposing $\dot{b}^a_i = 0$ once a stop by the robot is detected. Here, we are referring to stops at the network level, which can occur several time instants after the user releases the joypad. We detect such stops by searching for almost-identical consecutive network outputs. In our case, a stop is detected if $\|\mathbf{y}_i - \mathbf{y}_{i-1}\| < \tau_{stop}$, with $\tau_{stop} = 0.05$.

D Results

D.1 Training Details

We train the MANN architecture using the Adam optimizer with warm restart (AdamWR), configured as in [17] (i.e., $T_i = 10$, $T_{mul} = 2$, $\eta = 1.0 \cdot 10^{-4}$, $\lambda = 2.5 \cdot 10^{-3}$). Around 25 hours on an NVIDIA GeForce GTX 1650 GPU are required for 150 training epochs on the whole dataset, using mini-batches of 32 randomly selected samples. The Dropout keep probability is set to 0.7.

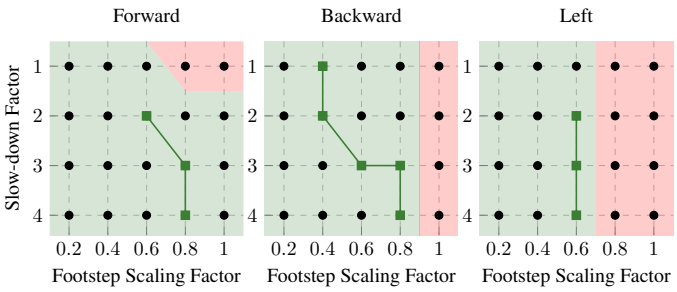

Figure 4: Simulated and experimental results for different combinations of slow-down and footstep scaling factors considered in our robustness analysis. Each figure refers to a different kind of walking motion.

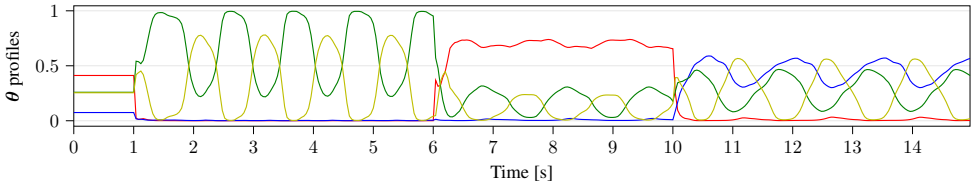

Figure 5: Blending coefficients $\boldsymbol{\theta}$ profiles with $K = 4$ experts for an articulated trajectory including standing (0-1s), straight (1-4s) and steered (4-6s) forward walking, right-side (6-10s), and left-side (10-15s) walking.

D.2 Robustness Analysis

We evaluate the robustness of ADHERENT to a challenging range of step sizes and walking speeds compatible with the locomotion capabilities of iCub. We carry out our analysis both in simulation and on the real robot. We apply a *slow-down factor* to the generated trajectories in the range $\{1, 2, 3, 4\}$. A *footstep scaling factor* in the range $\{0.2, 0.4, 0.6, 0.8, 1\}$ is instead applied to the footstep positions. The analysis is repeated for forward, backward, and left-side walking. Concerning simulations, Fig. 4 illustrates the results for all parameter combinations, with the successful trials and failures represented by the green and the red areas, respectively. For each considered motion, most combinations are successful. No matter the slow-down factor, the maximum admissible footstep scaling for successful backward and left-side walking is 0.8 and 0.6, respectively. As regards real experiments, the solid green line in Fig. 4 connects the most challenging parameter combinations resulting in successful outcomes. We observe that higher speed can be traded off with larger step sizes.

D.3 Blending Coefficients Activation

We analyze how the experts specialize in different motions by plotting the profiles of the corresponding blending coefficients $\boldsymbol{\theta}$ in Fig. 5. Note that $\boldsymbol{\theta}$ activations show distinctive periodic patterns characterizing each motion type. For instance, in both the straight and steered forward walking phases, only θ_1 (green) and θ_2 (yellow) are active, and specialize in left and right swing motions, respectively. Moreover, θ_3 (red) and θ_4 (blue) become active during right- and left-side walking, respectively. The real-time evolution of expert activations for different motions is shown in the supplementary video.

D.4 Human-likeness Analysis Details

In order to evaluate the *human likeness* of the trajectories executed using ADHERENT, we compare them with trajectories exploiting a fixed postural for the upper body. For both cases, Fig. 6 shows reference and measured joint trajectories for a representative set of upper-body joints during forward walking. When using a fixed postural, the measured joint positions oscillate in proximity of the constant reference. With the ADHERENT postural, the measured joint positions closely track the references produced by the network, despite the lower-level action of the *Whole Body QP Controller*. This demonstrates that reference motions produced by generators trained on human data can actually be realized on the real robot. Note that the learning-driven motion of the representative joints discussed above considerably contributes to the improvement in terms of human likeness that can be seen in the supplementary video.

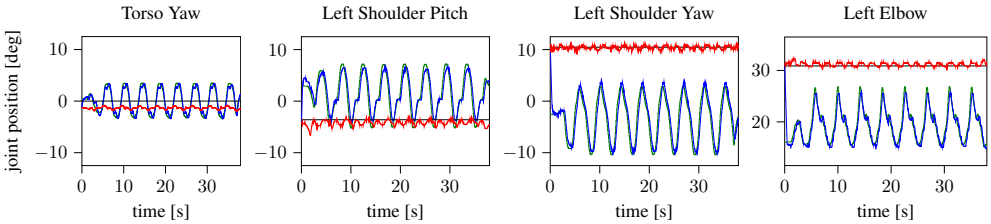

Figure 6: Joint trajectory control with ADHERENT postural vs. fixed postural for four representative upper-body joints. Green: ADHERENT postural reference trajectory. Blue: Measured trajectory with ADHERENT postural. Black: Fixed postural reference trajectory. Red: Measured trajectory with fixed postural.

E Future Work

In this work, we presented an implementation of the ADHERENT architecture employing instantaneous controllers for trajectory control. Possible future work includes investigating, comparing, and integrating ADHERENT with more advanced control architectures (i.e., MPC- or RL-based).

From a machine-learning perspective, ADHERENT must be retrained from scratch whenever new motion skills need to be added. This could be tackled by integrating recent continual/lifelong learning methods in the architecture.

Finally, an extension of our work for the navigation of uneven ground could be pursued by including perceptual terrain features in the network input. Enhancing the ADHERENT architecture with perception could indeed represent another significant step towards the development of general and human-like trajectory generation.

F Acknowledgments

D. Pucci acknowledges the support by the SoftManBot project under grant agreement No. 869855 and the An.Dy project under grant agreement No. 731540, which have received funding from the European Union's Horizon 2020 Research and Innovation Programme. L. Rosasco acknowledges the support by the European Research Council (grant SLING 819789), the Center for Brains, Minds and Machines, funded by NSF STC award CCF-1231216, the AFOSR projects FA9550-18-1-7009, FA9550-17-1-0390 and BAA-AFRL-AFOSR-2016-0007 (European Office of Aerospace Research and Development), the EU H2020-MSCA-RISE project NoMADS - DLV-777826, and the NVIDIA Corporation for GPUs donations.

References

[1] Siyuan Feng, Eric Whitman, X Xinjilefu, and Christopher G. Atkeson. Optimization based full body control for the atlas robot. In *Humanoids*, 2014.

[2] Giulio Romualdi, Stefano Dafarra, Yue Hu, and Daniele Pucci. A Benchmarking of DCM Based Architectures for Position and Velocity Controlled Walking of Humanoid Robots. In *Humanoids*, 2018.

[3] Stefano Dafarra, Gabriele Nava, Marie Charbonneau, Nuno Guedelha, Francisco Andrade, Silvio Traversaro, Luca Fiorio, Francesco Romano, Francesco Nori, Giorgio Metta, and Daniele Pucci. A Control Architecture with Online Predictive Planning for Position and Torque Controlled Walking of Humanoid Robots. In *IROS*, 2018.

[4] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: Modelling, Planning and Control*. Springer, 3rd edition edition, 2008.

[5] Tan-Viet-Anh Truong, David Flavigne, Julien Pettrée, Katja Mombaur, and Jean-Paul Laumond. Reactive synthesizing of human locomotion combining nonholonomic and holonomic behaviors. In *BioRob*, 2010.

[6] Hongkai Dai, Andrés Valenzuela, and Russ Tedrake. Whole-body motion planning with centroidal dynamics and full kinematics. In *Humanoids*, 2014.

[7] Alexander Herzog, Nicholas Rotella, Stefan Schaal, and Ludovic Righetti. Trajectory generation for multi-contact momentum control. In *Humanoids*, 2015.

[8] Justin Carpentier, Steve Tonneau, Maximilien Naveau, Olivier Stasse, and Nicolas Mansard. A versatile and efficient pattern generator for generalized legged locomotion. In *ICRA*, 2016.

[9] George Mesesan, Johannes Englsberger, Gianluca Garofalo, Christian Ott, and Alin Albu-Schäffer. Dynamic Walking on Compliant and Uneven Terrain using DCM and Passivity-based Whole-body Control. In *Humanoids*, 2019.

[10] N Ramuzat, G Buondonno, S Boria, and Olivier Stasse. Comparison of Position and Torque Whole Body Control Schemes on the TALOS Humanoid Robot. 2021.

[11] Michael Bombile and Aude Billard. Capture-point based balance and reactive omnidirectional walking controller. In *Humanoids*, 2017.

[12] Milad Shafiee, Giulio Romualdi, Stefano Dafarra, Francisco Javier Andrade Chavez, and Daniele Pucci. Online DCM Trajectory Generation for Push Recovery of Torque-Controlled Humanoid Robots. In *Humanoids*, 2019.

[13] Katja Mombaur, Anh Truong, and Jean-Paul Laumond. From human to humanoid locomotion—an inverse optimal control approach. *Autonomous Robots*, 2010.

[14] Alessandro Vittorio Papadopoulos, Luca Bascetta, and Gianni Ferretti. Generation of human walking paths. In *IROS*, 2013.

[15] Isabelle Maroger, Noelie Ramuzat, Olivier Stasse, and Bruno Watier. Human Trajectory Prediction Model and Its Coupling With a Walking Pattern Generator of a Humanoid Robot. *IEEE Robotics and Automation Letters*, 2021.

[16] Daniel Holden, Taku Komura, and Jun Saito. Phase-functioned neural networks for character control. *ACM Transactions on Graphics*, 2017.

[17] He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. Mode-adaptive Neural Networks for Quadruped Motion Control. *ACM Transactions on Graphics*, 2018.

[18] Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. DReCon: data-driven responsive control of physics-based characters. *ACM Transactions on Graphics*, 2019.

[19] Daniel Holden, Oussama Kanoun, Maksym Perepichka, and Tiberiu Popa. Learned motion matching. *ACM Transactions on Graphics*, 2020.

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

[21] Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. Learning Agile Robotic Locomotion Skills by Imitating Animals. In *Robotics: Science and Systems*, 2020.

[22] Giorgio Metta, Lorenzo Natale, Francesco Nori, Giulio Sandini, David Vernon, Luciano Fadiga, Claes von Hofsten, Kerstin Rosander, Manuel Lopes, José Santos-Victor, Alexandre Bernardino, and Luis Montesano. The iCub humanoid robot: An open-systems platform for research in cognitive development. *Neural Networks*, 2010.

[23] N.S. Pollard, J.K. Hodgins, M.J. Riley, and C.G. Atkeson. Adapting human motion for the control of a humanoid robot. In *ICRA*, 2002.

[24] Christian Ott, Dongheui Lee, and Yoshihiko Nakamura. Motion capture based human motion recognition and imitation by direct marker control. In *Humanoids*, 2008.

[25] Kanako Miura, Mitsuharu Morisawa, Shin'ichiro Nakaoka, Fumio Kanehiro, Kensuke Harada, Kenji Kaneko, and Shuuji Kajita. Robot motion remix based on motion capture data towards human-like locomotion of humanoid robots. In *Humanoids*, 2009.

[26] L. Penco, B. Clement, V. Modugno, E. Mingo Hoffman, G. Nava, D. Pucci, Nikos G. Tsagarakis, J. B. Mouret, and S. Ivaldi. Robust Real-Time Whole-Body Motion Retargeting from Human to Humanoid. In *Humanoids*, 2018.

[27] Kourosh Darvish, Yeshasvi Tirupachuri, Giulio Romualdi, Lorenzo Rapetti, Diego Ferigo, Francisco Javier Andrade Chavez, and Daniele Pucci. Whole-Body Geometric Retargeting for Humanoid Robots. *Humanoids*, 2019.

[28] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive Mixtures of Local Experts. *Neural Computation*, 1991.

[29] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa. The 3D linear inverted pendulum mode: a simple modeling for a biped walking pattern generation. *IROS*, 2001.